Masteroppgave

NTNU
Norges teknisk-naturvitenskapelige universitet
Fakultet for informasjonsteknologi og elektroteknikk
Institutt for matematiske fag

Mathilde Tangen Hollås

Programmering i sannsynlighet

En kvalitativ innholdsanalyse om hvordan programmering er integrert inn i sannsynlighet i lærebøker for 9. trinn

Masteroppgave i MLREAL
Veileder: Alexander Schmeding
Juni 2024

NTNU
Kunnskap for en bedre verden

Mathilde Tangen Hollås

Programmering i sannsynlighet

En kvalitativ innholdsanalyse om hvordan programmering er integrert inn i sannsynlighet i lærebøker for 9. trinn

Masteroppgave i MLREAL
Veileder: Alexander Schmeding
Juni 2024

Norges teknisk-naturvitenskapelige universitet
Fakultet for informasjonsteknologi og elektroteknikk
Institutt for matematiske fag

NTNU

Kunnskap for en bedre verden

Sammendrag

Denne studien har undersøkt programmeringsinnholdet i sannsynlighet i lærebøker for 9. trinn. Formålet har vært å få innsikt i hvordan programmering har blitt innført i sannsynlighetstemaet og hvordan programmering og sannsynlighet knyttes sammen. Ved å få et innblikk inn i lærebøkenes tilnærminger vil det kunne øke lærernes bevissthet omkring bruken av programmering i sannsynlighet og bøkenes programmeringsinnhold i temaet. Studiens forskningsspørsmål er: (*1) Hva karakteriserer programmeringsinnholdet i sannsynlighetstemaet i norske matematikkbøker for 9. trinn?* og *(2) På hvilke måter knytter lærebøkene programmering og sannsynlighet sammen?*

Studien har benyttet seg av metoden kvalitativ innholdsanalyse og har analysert oppgaver og eksempler i sannsynlighetskapitlene i to lærebøker for 9. trinn. Datamaterialet ble analysert ved hjelp av rammeverket til Bråting og Kilhamn (2021) om handlinger og begreper for å analysere bruken av programmering i lærebøkene. I tillegg ble en analyse om læring av sannsynlighet, utviklet fra Castro (1998) og Wilensky (1995) sine rammeverk om undervisning i sannsynlighet, utført på datamaterialet for å undersøke hvordan programmering blir brukt for å utvikle elevenes sannsynlighetsforståelse.

Programmeringsinnholdet i sannsynlighetstemaet i norske matematikkbøker for 9. trinn karakteriseres av at eleven skal enten kopiere, utvide eller lage en egen kode ved bruk av tekstprogrammering i Python, samt en mangel på problemer med feilsøking. Dataene viser at programmeringsoppgavene knytter programmering og sannsynlighet sammen ved å benytte programmering hovedsakelig som et verktøy for simulering og beregning, der kunnskaper og ferdigheter innenfor sannsynlighet holdes sentralt i lærebøkene. Ved å være bevisst på lærebøkenes styrker, mangler og forskjeller i innhold og struktur, som denne studien har belyst i sannsynlighetskapitlene for 9. trinn, kan lærere få et innblikk inn i lærebøkene og vurdere og tilpasse til deres undervisning.

Nøkkelord: lærebøker i matematikk, lærebokanalyse, programmering, sannsynlighet

Abstract

This study has examined the programming content in probability in 9th grade textbooks. The purpose has been to gain insight into how programming has been introduced in probability and how programming and probability are linked. By gaining an insight into the approaches in the textbooks, it could increase teachers' awareness of the use of programming in probability and the programming content of the books in this topic. The research questions of the study are: *(1) What characterizes the programming content in the topic probability in Norwegian mathematics textbooks for 9th grade?* and *(2) In what ways do the textbooks connect programming and probability?*

The study has employed the method of qualitative content analysis and has analyzed tasks and examples in the probability chapters of two 9th grade textbooks. The data were analyzed using the framework by Bråting and Kilhamn (2021) about actions and concepts to analyze the use of programming in the textbooks. In addition, an analysis on learning probability, developed from Castro (1998) and Wilensky (1995) frameworks on teaching probability, was conducted on the data to examine how programming is used to develop students' understanding of probability.

In both textbooks, a lack of debugging was observed, which is an important concept in programming. It seems that mathematical knowledge is central in the data, where programming is mainly used as a tool for simulation and calculation. From the findings of the study, there is a distinct difference in types of tasks and information provided to students in the different textbooks. Whether decomposed problems with many subtasks or a more open-ended problem are most favorable in learning probability and programming, or other topics, is an unanswered question that could be interesting to explore further.

The programming content in the topic probability in Norwegian mathematics textbooks for 9th grade is characterized by students either copying, extending, or creating their own code using text-based programming in Python, along with a lack of debugging problems. The data shows that programming tasks link programming and probability by using programming mainly as a tool for simulation and calculation, where knowledge and skills in probability are kept central in the textbooks. By being aware of the textbooks' strengths, weaknesses, and differences in content and structure, as highlighted by this study in the probability chapters for 9th grade, teachers can gain insight into the textbooks and assess and adapt to suit their teaching.

Keywords: mathematics textbooks, textbook analysis, programming, probability

Forord

Denne masteroppgaven markerer slutten av min lektorutdanning i realfag på NTNU og min tid i Trondheim. Det har vært 5 år med læring, både skole- og hjemmeeksamener, nye venner og opplevelser, der jeg vil avslutte Trondheimseventyret med mange gode minner og stunder. Masteroppgaven har vært utfordrende, men utrolig spennende å arbeide med og er et lærerikt prosjekt jeg er veldig stolt av. Det er ingen tvil om at jeg vil bruke min kunnskap fra masteren inn i mitt fremtidige arbeid som lektor.

Selv om det er mitt navn som står på denne oppgaven, er det flere som har bidratt til at jeg kom i mål som jeg vil takke.

Jeg vil takke min veileder, Alexander Schmeding, for nyttige diskusjoner og tilbakemeldinger gjennom dette halvåret. Du har alltid vært tilgjengelig og gjort arbeidet med masteren til en lærerik og interessant fordypning inn i lærebøkene.

Takk til familie og venner, som har gjort det mulig å kose seg og ta pauser samtidig som man stresser over øvinger, innleveringer, eksamener og, selvfølgelig, masteren. Både dette halvåret og gjennom hele studieløpet har vi funnet på mye kos, og jeg hadde ikke kommet meg gjennom uten dere.

Til slutt må jeg takke teitingen i mitt liv, Øyvind. På 35 kvadrat har vi levd i 5 år, og klart å holde sammen. Både veldig nære grunnet Covid og langt unna grunnet utveksling. Selv dette semesteret kom vi oss gjennom med mastere, kjøp og salg av bolig, bryllupsplaner og mer, og det kan jeg si med sikkerhet er fordi jeg har deg. Nå er vi begge i mål og kan begynne et nytt (voksen) kapittel sammen.

Juni, 2024

Mathilde

Innhold

Figurer

Tabeller

1 Innledning

1.1 Bakgrunn for studien

Samfunnet rundt oss blir i økende grad mer digitalisert, som gjør digital kompetanse til en nødvendig kunnskap for alle borgere i den digitale verdenen vi lever i. Den digitale utviklingen har påvirket alle samfunnsområder og økonomien, med en stadig større innvirkning på hverdagslivet (Bocconi et al., 2022, s. 14). Etter COVID-19 pandemien akselererte bruken av digitale ressurser innenfor utdanning, men førte i tillegg til en forsterkning av eksisterende utfordringer og ulikheter mellom tilgang og kompetanse til digitale teknologier. Både før og etter pandemien har det vært en økende forståelse for at digital kompetanse inneholder mer enn å mestre grunnleggende digitale ferdigheter, som skaper et behov for å bedre forstå algoritmisk tenkning (Computational Thinking) og hvordan denne tankegangen kan bidra til digitale ferdigheter og kompetanse.

Algoritmisk tenkning har de siste årene blitt sett på som en grunnleggende ferdighet for alle, ikke bare for teknologibransjen. Flere og flere land fremmer å utvikle ferdigheter innenfor algoritmisk tenkning ved å implementere programmering inn i læreplanene, som Norge gjorde med introduksjonen av programmering inn i den nye læreplanen utgitt i 2020 (Bocconi et al., 2022). I 2021 lanserte EU en plan som vil vare til 2027 om å støtte integrasjonen av digital kompetanse, og dermed algoritmisk tenkning, i utdanningen i forskjellige medlemsland (Bocconi et al., 2022, s. 8). Algoritmisk tenkning er ikke lenger en trend som vil forsvinne. Mens denne prosessen fortsetter å utvikle seg, er det viktig å overvåke og vurdere implementeringen av algoritmisk tenkning i læreplanene for å undersøke og sikre elevenes utbytte.

Den faglige plasseringen av algoritmisk tenkning og programmering varierer fra land til land, med tre ulike tilnærminger (Bocconi et al., 2022, s. 6). Noen land velger å legge det som et tverrfaglig tema. Andre introduserer det som en del av et eget fag og den siste tilnærmingen er å plassere algoritmisk tenkning inn i andre eksisterende fag. I Norge er sistnevnte tilnærming gjort ved å innlemme det inn i matematikk, naturfag, musikk og kunst og håndverk, der hovedansvaret har blitt henlagt til matematikkfaget.

Siden innføringen av den nye læreplanen, programmering og algoritmisk tenkning som en ferdighet, har det dukket opp flere utfordringer som for eksempel prioriteringer av læreplanmål innad i fag, vurdering av elevenes kompetanse og mangel på tilstrekkelig kompetanse hos lærere. I emnet RFEL3100 på NTNU undersøkte jeg hvordan lærere opplever å undervise om programmering i matematikkfaget, og hvordan læreboka ble brukt som en ressurs (Hollås, 2023). I intervjuer av lærere kom det frem at undervisning med programmering ble nedprioritert grunnet lærernes lave programmeringskompetanse og konkurranse med matematiske temaer som tradisjonelt har blitt undervist i faget. Lærernes lave kompetanse førte til en utrygghet i undervisningen av programmering der de ikke benyttet seg av lærebøkene som en ressurs da nivået på programmering i bøkene ble for høyt.

Lærebøker er en betydelig økonomisk investering og påvirker sterkt hva elevene lærer da det er vanlig for lærere å forholde seg til rekkefølgen og innholdet i fagets lærebok i undervisningen (Reys et al., 2004, s. 61). I tillegg til å presentere matematiske temaer i en bestemt rekkefølge, antyder også lærebøkene innholdet lærerne bør undervise i. Man kan se på lærebøkene som en ressurs for undervisningsplaner komplett med eksempelproblemer, figurer og diagrammer, eksempler med løsningsforslag og oppgaver til arbeid og lekser. Lærebøkene kan dermed bidra til eller begrense forståelsen elevene vil kunne tilegne seg. De fleste forlag vurderer ikke effektiviteten eller kvaliteten til lærebøkene før utgivelse eller mens de blir brukt i undervisning. Dette resulterer i at lærebøker, som er en stor del av elevenes læring, generelt gjennomgår få endringer fra en utgave til den neste, og vil bare skje hvis læreplanen endrer seg betydelig. I Norge eksisterer det ingen sentral kvalitetskontroll for skolebøker. Det bør dermed skje en vurdering av innføringen av algoritmisk tenkning og programmering både før og etter publisering av matematikklærebøker som følger den nye læreplanen. Dette kan gi innsikt i hvordan faget undervises som en konsekvens, hvordan elevenes læring påvirkes, samt om elevene har utviklet ferdigheter i algoritmisk tenkning (Bocconi et al., 2022, s. 88).

1.2 Forskningsspørsmål og undersøkelsens formål

Siden programmering og algoritmisk tenkning er nye momenter i matematikkfaget er det viktig å vurdere hvordan de blir fremstilt i lærebøkene, blant annet for å få kjennskap til hvordan lærebøkene kan være en ressurs for lærere i undervisningen og hvordan dette samsvarer med læreplanen (Kunnskapsdepartementet, 2018). Et av kompetansemålene som eksplisitt nevner programmering er fra 9. trinn, som sier at eleven skal «simulere utfall i tilfeldige forsøk og beregne sannsynligheten for at noe skal inntreffe, ved å bruke programmering» (Kunnskapsdepartementet, 2020a). Jeg ønsket derfor å undersøke sannsynlighet i lærebøkene for 9. trinn og se hvordan programmering og algoritmisk tenkning har blitt integrert inn i sannsynlighet siden det spesifikt blir henvist til programmering i læreplanmålet for dette temaet.

I Sverige ble det gjort en studie av Bråting og Kilhamn (2021) der de analyserte det matematiske innholdet i de nye svenske lærebøkene etter innføringen av deres nye læreplan som også inneholder algoritmisk tenkning og programmering. I min studie har jeg basert meg på rammeverket utviklet av Bråting og Kilhamn (2021), da vi har liknende forskningsspørsmål og undersøker programmeringsinnholdet i lærebøker. Rammeverket, og hvordan det ble utviklet, blir presentert i Kapittel 2.2. For å undersøke programmeringsinnholdet i sannsynlighetskapitlene i lærebøker for 9. trinn stiller jeg nesten de samme spørsmålene som Bråting og Kilhamn (2021), men med en innsnevring mot sannsynlighet. Det finnes, per nå, ingen liknende undersøkelse innenfor sannsynlighet og klassetrinnet i Norge eller i Sverige. Fra mitt perspektiv er det dermed et hull i forskningen på dette feltet som jeg vil svare på ved hjelp av spørsmålene:

1) Hva karakteriserer programmeringsinnholdet i sannsynlighetstemaet i norske matematikkbøker for 9. trinn?
2) På hvilke måter knytter programmeringsoppgavene programmering og sannsynlighet sammen?

Jeg vil poengtere at forskningsspørsmål 2 handler om sammenhengen mellom programmering og matematikk og hvordan de blir brukt i oppgavene, mens forskningsspørsmål 1 har algoritmisk tenkning og programmering i fokus.

Formålet med studien er å bidra til mer kunnskap om integrasjonen av algoritmisk tenkning og programmering i sannsynlighetstemaet i matematikklærebøker for 9. trinn siden dette er nytt innhold i bøkene og undervisningen etter innførelsen av den nye læreplanen. Dette kan gi et innblikk inn i forventninger til programmeringskunnskap hos elever og lærere og øke bevisstheten rundt lærebøkenes styrker og begrensninger i læringen av programmering. Ved å ha god kjennskap til innholdet i lærebøkene kan min studie hjelpe lærere å gjøre ulike tilpasninger i programmeringsoppgaver for å tilrettelegge for best mulig undervisning.

Ved å undersøke programmeringsinnholdet i lærebøkene vil det kunne føre til bedre undervisning av sannsynlighet programmering og algoritmisk tenkning. Ifølge FNs fjerde bærekraftsmål om god utdanning skal man arbeide for å «sikre inkluderende, rettferdig og god utdanning og fremme muligheter for livslang læring for alle» (FN-sambandet, 2024). Digital kompetanse er og vil bli en viktig kunnskap å tilegne elever for å fremme deres muligheter og deltakelse i det moderne samfunn. Dermed er det viktig å undersøke lærebøker elevene vil møte og benytte seg av i sin skolegang og sørge for at de får gode ressurser rundt seg for å tilegne seg ferdigheter innenfor programmering og algoritmisk tenkning.

1.3 Oppgavens oppbygning

Videre i Kapittel 2 vil jeg definere algoritmisk tenkning og redegjøre for de teoretiske rammeverkene benyttet i denne studien: Bråting og Kilhamn (2021) sine handlinger og begreper og en egenutviklet modell for læring av sannsynlighet basert på rammeverkene til Castro (1998) og Wilensky (1995), samt programmeringens muligheter i sannsynlighetsundervisning. I Kapittel 3 vil jeg beskrive metodologien som ligger til grunn for studien, før Kapittel 4 gir en beskrivelse og presentasjon av analyseprosessen og studiens funn med eksempler fra datamaterialet. Oppgaven avsluttes med en diskusjon opp mot relevant teori av funnene i analysen i Kapittel 5, og konklusjon i Kapittel 6, samt eventuell videre forskning.

2 Teori

I dette kapittelet følger en presentasjon av de teoretiske rammeverkene brukt som analyseverktøy og som teoretisk ramme for teksten. Ulike programmeringsgrener og begrepet algoritmisk tenkning vil bli definert i Kapittel 2.1, da algoritmisk tenkning og sannsynlighet er sentrale temaer for denne studien. I Kapittel 2.2 vil ulike rammeverk for algoritmisk tenkning og programmering i matematikk presenteres, samt endringer gjort og hvorfor jeg anser det som hensiktsmessig å bruke i denne studien. Videre vil jeg presentere hvordan temaet sannsynlighet blir vektlagt i læreplanen i Kapittel 2.3, og redegjøre for egenutviklet modell om læring av sannsynlighet ut fra to eksisterende rammeverk for sannsynlighetsinnholdet i matematikkoppgaver i Kapittel 2.4. Til slutt vil det i Kapittel 2.5 presenteres teori som knytter sammen programmering og sannsynlighet.

2.1 Algoritmisk tenkning

I tritt med datamaskinenes og den digitale verdens utvikling, ble det også utviklet ulike programmeringsspråk som gjør oss i stand til å bruke datamaskinen slik vi gjør i dag. Programmering i seg selv er å beskrive, eller «kode», en serie instruksjoner som må utføres for å oppnå et bestemt resultat (Vihovde, 2023). Programmet skal inneholde presise instruksjoner for hva datamaskinen skal gjøre, samt hvordan maskinen skal reagere på ulike hendelser eller input som kan oppstå under kjøringen av programmet. Skolen opplevde en programmeringsrenessanse etter publiseringen av Wing (2006) sin artikkel om algoritmisk tenkning og man så en økt nytteverdi for denne ferdigheten i et mer og mer digitalisert samfunn.

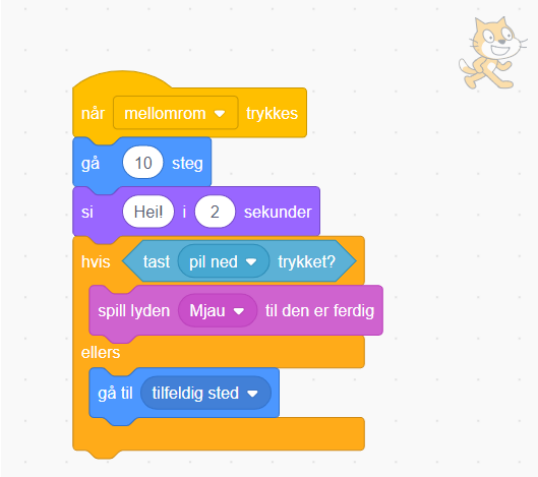
I undervisningssammenheng for eldre elever har Python blitt et populært tekstbasert programmeringsspråk som er fleksibelt, gratis og lett å lære med utallige ressurser liggende ute. Python er ofte anbefalt til nybegynnere grunnet språkets enkle syntaks og innebygde funksjoner og biblioteker som kan importeres og kalles på, og er i tillegg mye brukt i industrien (Lær Kidsa Koding, u.å.). Men, grunnet at datamaskinen trenger presise stegvise instruksjoner kan det ofte føre til feil i koden i form av syntaks- og skrivefeil og feil logisk tankegang for nybegynnere.

```
tall = int(input("Skriv et tall: "))
svar = 3 + tall
print(svar)
```

Figur 2.1. Eksempel på tekstprogrammering med bruk av Python. Hentet fra kodeklubben.no

Fokuset på programmering har også ført til utviklingen av flere blokkprogrammeringsspråk, slik at barn og unge lettere kan lære seg programmering uten å bli hindret av feilene som er vanlige å oppleve i tekstprogrammering. I blokkprogrammering er skrivefeil umulig da man benytter ferdige blokker med ulike funksjoner som legges etter hverandre og kombineres (Kivle, u.å.).

Men selv om man eliminerer skrive- og syntaksfeil, er det ingen garanti for at programmet fungerer slik man ønsker. Blokkene må kombineres i en viss rekkefølge for et visst resultat og man får utviklet sin forståelse for å tenke med stegvise instruksjoner som en datamaskin. Programmeringplattformene Scratch og Micro:Bit er mye brukt i undervisning med blokkprogrammering.



Figur 2.2. Eksempel på blokkprogrammering i Scratch. Hentet fra skolekoding.no

Å programmere uten datamaskin kan være med på å lære elevene sammenhenger innen programmering og få en større forståelse for begreper og hvordan en datamaskin svarer på forskjellige kommandoer (Statped, 2021). Ved å først jobbe analogt med forståelsen av hva som skjer i en kode, og bli vant til å tenke algoritmisk, kan det gjøre introduksjonen av den digitale programmeringen enklere. Et eksempel på analog programmering er å lage en algoritme med stegvise instruksjoner for hvordan man skal bevege seg i et rutenett på gulvet. Da vil man lære mulige fremgangsmåter for å omgjøre virkelige handlinger til instruksjoner.

I nyere tider har algoritmisk tenkning og programmering blitt sentralt i debatten om å utvikle elevers IKT-ferdigheter, der disse ferdighetene, som ofte blir omtalt som *21st century skills* i litteraturen, anses av mange som like grunnleggende som regning og lesing (Bocconi et al., 2018, s. 1). Fra litteraturen og forskjellige land finnes det ingen felles definisjon på hva algoritmisk tenkning innebærer, men i de nordiske landene er hovedforståelsen lik og fokuserer på to momenter: problemløsning og digital kompetanse.

Begrepet algoritmisk tenkning ble introdusert med innføringen av Kunnskapsløftet 2020 (LK20) og står spesifikt nevnt i kjerneelementet *utforskning og problemløsning* (Kunnskapsdepartementent, 2020a). Utdanningsdirektoratet (Udir) ser dermed på algoritmisk tenkning som en problemløsningsmetode ut fra denne plasseringen. Dette er dermed et nytt begrep som skal følge elevene gjennom hele grunnskolen. Ifølge Utdanningsdirektoratet innebærer algoritmisk tenkning:

> Å tenke algoritmisk er å vurdere hvilke steg som skal til for å løse et problem, og å kunne bruke sin teknologiske kompetanse for å få en datamaskin til å løse (deler av) problemet. I dette ligger også en forståelse av hva slags problemer/oppgaver som kan løses med teknologi og hva som bør overlates til mennesker. (Utdanningsdirektoratet, 2019a)

Internasjonalt brukes begrepet *computational thinking* (CT), som på norsk er oversatt til algoritmisk tenkning. Det finnes mange ulike definisjoner på hva CT er, men ifølge Wing (2006) innebærer det å løse problemer, designe systemer, tenke rekursivt, kunne forenkle utfordrende problemer til noe man kan løse, og bruke abstraksjon og dekomponering når man angriper en stor og kompleks oppgave. Det er mulig å identifisere et sett med kjernekonsepter under CT-paraplyen, som *abstraksjon*, *algoritmisk tenkning*, *automatisering*, *dekomponering* og *generalisering* (Bocconi et al., 2016; referert i Bocconi et al., 2018, s. 7). Det er viktig å poengtere at algoritmisk tenkning her ikke betyr det samme som CT. Internasjonalt har de et trangere syn på hva algoritmisk tenkning innebærer, der det er et konsept innenfor CT og ikke omfavner alle sidene ved begrepet CT. Disse kjernekonseptene er igjen relatert til et sett med arbeidsmåter, som inkluderer *å lage program*, *testing og feilsøking*, *samarbeid og kreativitet*, og *evnen til å håndtere åpne problemer*. Med denne forståelsen kan vi se på CT som en grunnleggende kompetanse for å bli en informert medborger som er i stand til å takle samfunnsutfordringer (Bocconi et al., 2018, s. 7). Ifølge Wing (2006) kan konsepter innenfor CT bli brukt til å tilnærme og løse problemer, styre hverdagen og kommunisere og samhandle med andre mennesker, og dermed være en ferdighet for alle, uansett hvilket yrke eller del av livet man befinner seg i. I resten av teksten vil algoritmisk tenkning henvise til computational thinking og dens definisjon, der jeg vil benytte meg av forkortelsen CT og algoritmisk tenkning om hverandre.

2.2 Modell for analyse

I denne studien har jeg tatt utgangspunkt i Bråting og Kilhamn (2021) sitt rammeverk for å se på programmeringsinnholdet i lærebøker for matematikk. Jeg vil først presentere rammeverkene Bråting og Kilhamn (2021) har utviklet sitt analyseverktøy og rammeverk fra, som er *5E-modellen* til Benton et al. (2016) og *CT-modellen* til Brennan og Resnick (2012), før jeg går videre til Bråting og Kilhamn (2021) sine *handlinger* og *begreper*. Til slutt vil jeg gi en begrunnelse for bruken av rammeverket samt mine endringer.

2.2.1 De fem E'er

Som i Norge, har England og Sverige innført programmering inn i skolen, og flere rammeverk som omhandler algoritmisk tenkning har blitt utviklet (Bocconi et al., 2018). Det ble dermed satt i gang et prosjekt kalt ScratchMath, som var et treårig prosjekt utført i Storbritannia om hvordan man kan bygge matematisk kunnskap gjennom programmering i Scratch (Benton et al., 2016). Scratch er et gratis blokkbasert kodespråk med et enkelt visuelt grensesnitt som lar barn og unge lage digitale historier, spill og animasjoner (Scratch, u.å.). Gjennom ScratchMath ble det utviklet et rammeverk kalt *5E* som har som fokus å gi pedagogiske strategier for å designe og evaluere læring av algoritmisk tenkning og utnyttelsen av den for å forbedre matematisk engasjement og forståelse med en konstruktivistisk tilnærming (Benton et al., 2016; Benton et al., 2017).

Benton et al. (2016) sin 5E-modell skal ikke forveksles med BSCS 5E instruksjons-modellen som er primært rettet mot naturvitenskapene (Bybee et al., 2006). Selv om begge modellene består av fem prinsipper og vektlegger elevengasjement og dybdelæring, er 5E (Benton et al., 2016) mer tilpasningsdyktig og fleksibel til forskjellige disipliner mens BSCS 5E (Bybee et al., 2006) er mer begrenset mot naturvitenskapene grunnet sine mer teoretiske prinsipper. 5E (Benton et al., 2016) er i tillegg mer rettet

mot å integrere teknologi inn i undervisningen da blokkprogrammering var et fokuspunkt i ScratchMath prosjektet. I min studie blir lærebøker med tekstbasert programmering analysert, som er en annen type arbeidsmåte enn ved å bruke blokkprogrammering, men inneholder de samme konseptene.

5E-modellen består av fem prinsipper som kan veilede utformingen av oppgaver og læringsaktiviteter for å fremme læring (Benton et al., 2016). Disse prinsippene omfatter ulike handlingsaspekter som kan kombineres på ulike måter for å utfordre og gi nye innfallsvinkler til elevene på oppgavene som skal løses. De fem prinsippene er: *Explore [Utforske]*, *Explain [Forklare]*, *Envisage [Forestille seg]*, *Exchange [Dele]* og *BridgE [Knytte sammen]*. Videre vil jeg gi en beskrivelse av disse prinsippene der Benton et al. (2016) er brukt som hovedkilde.

Explore [Utforske]

Med en konstruktivistisk tilnærming, der det forutsettes at de som lærer må konstruere kunnskapen selv, verdsettes læring gjennom å gi muligheter til å utforske måter å håndtere ulike begrensninger og innfallsvinkler ved å bruke ferdigheter som iterativ tenkning, problemløsning og kreativitet. Elever bør derfor få muligheter til å undersøke ideer gjennom utforskning og feilsøking, slik at de kan få et innblikk i muligheter og begrensninger som følger med ulike verktøy og hjelpemidler (Benton et al, 2017). En del av dette prinsippet er å lære elevene hvordan man kan, ifølge Benton et al. (2016), «ta kontroll over sin egen læring» og finne ut hvorfor man får forskjellige resultater (s. 5).

Explain [Forklare]

Ved å kunne forklare hva man har lært og hvorfor man har valgt en metode kan man oppnå en større og klarere forståelse for ulike ideer. Flere forskere har fremhevet den kognitive fordelen av muntlige forklaringer og refleksjoner, både med medelever og i klassediskusjoner. Denne refleksjonen over egen tankegang og kunnskap er en nøkkelkomponent i den konstruktivistiske tilnærmingen, der programmeringsspråket blir verktøyet man tenker med. Elevene skal dermed kunne forklare gitt eller egen kode og dets tiltenkte utfall trinn for trinn for å styrke refleksjonen over egen tenkning. Programmeringsspråk har i tillegg flere ord og uttrykk med betydning forskjellig fra elevenes vanlige språkbruk som er viktig at de har kjennskap til for at de skal lære å kunne forklare hva som skjer i en kode.

Envisage [Forestille seg]

I programmering, og i algoritmisk tenkning, er det viktig å ha et mål i tankene og prøve å se for seg hva resultatet kan være før man prøver det ut. Nybegynnervennlige programmeringsverktøy er ofte innebygd til å finne syntaksfeil, og elevene trenger kun å feilsøke når de har et klart mål med koden. Med andre ord er det enkelt å generere et utfall, men ikke nødvendigvis det spesifikke utfallet man var ute etter. Derfor burde elevene oppmuntres til å forutsi resultater før det skjer og deretter reflektere over resultatet. Det er viktig å balansere dette prinsippet med *Explore [Utforske]*, for å kunne gi muligheter til utforskning som tillater elevene å oppdage ideer samt anledninger til å forestille seg utfallet først.

Exchange [Dele]

Ved å jobbe sammen og dele ideer får man muligheten til å se et problem fra et annet perspektiv, samt å forsvare egen metode og sammenlikne den med andres. Ved å høre om andres ideer kan det potensielt resultere i endringer i egen tankeprosess, og kan være spesielt nyttig i oppklaring av misoppfatninger eller forklare ideer som ikke har festet rot. Derimot er elevers samarbeidsevne under utvikling og de kan trenge hjelp til å arbeide sammen, løse uenigheter og å vite hva slags spørsmål man burde stille hverandre. Dermed handler dette prinsippet om å inkludere muligheter for å forklare og bygge på hverandres strategier gjennom samarbeid med medelever og plenumsdiskusjoner.

BridgE [Knytte sammen]

I en godt designet konstruktivistisk aktivitet burde kraftige ideer være en del av aktiviteten. Om en idé er kraftig vil bli gitt av dens kobling til andre fag, som matematikk, og språket som blir brukt. Med andre ord må ideen kobles, re-kontekstualiseres og gjenoppbygges innenfor matematikk og med matematisk språk. Dermed er det viktig i dette prinsippet å eksplisitt synliggjøre koblingen mellom programmering og matematiske ideer for elevene.

2.2.2 Nøkkeldimensjoner for algoritmisk tenkning

Brennan og Resnick (2012) har utviklet et rammeverk for algoritmisk tenkning ut fra tidligere studier de har utført ved bruk av verktøyet Scratch. Rammeverket består av tre nøkkeldimensjoner: *begreper innenfor algoritmisk tenkning* (begrepene designere engasjerer seg imens de programmerer, som iterasjon, løkker, etc.), *praksis for algoritmisk tenkning* (praksis designerne utvikler etter hvert som de engasjerer seg i konseptene, som å feilsøke prosjekter eller modifisere andres arbeid), og *perspektiver i algoritmisk tenkning* (perspektivene designere danner om verden rundt dem og om seg selv). Videre gir jeg en kort beskrivelse av hva disse dimensjonene innebærer.

Begreper innenfor algoritmisk tenkning

Brennan og Resnick (2012) har identifisert syv begreper de anser som svært nyttig i Scratch, og som kan overføres til tekstbasert programmering. Videre kommer en kort beskrivelse av hvert begrep.

- *Sekvenser*: En sekvens av programmeringsinstruksjoner som spesifiserer handlingen som skal utføres.
- *Løkker*: En løkke kan brukes til å uttrykke en repeterende sekvens av instruksjoner mer kortfattet. F.eks.: for-løkke, while-løkke.
- *Hendelser*: En hendelse får en annen hendelse til å skje. F.eks.: input.
- *Parallellisme*: Sekvenser av instruksjoner som skjer samtidig.
- *Betingelse*: Evnen til å ta beslutninger basert på gitte betingelser, som gir muligheten for flere utfall. F.eks.: if-setninger.
- *Operatorer*: Operasjoner som representerer en spesifikk matematisk eller logisk handling, og muliggjør utførelsen av numeriske og strengmanipulasjoner. F.eks.: % (modulo regning i Python), and/or.
- *Data*: Data involverer å lagre, hente og oppdatere verdier. F.eks.: variabler og lister.

Praksis for algoritmisk tenkning

Denne dimensjonen fokuserer på tanke- og læringsprosessen, der man går forbi *hva* man lærer og videre til *hvordan* man lærer i oppgaver med algoritmisk tenkning (Brennan & Resnick, 2012, s. 7). Brennan og Resnick (2012) endte opp med fire praksiser som beskrives kort under.

- *Være inkrementell og iterativ*: Adaptiv prosess der man går frem mot en løsning med små steg.
- *Testing og feilsøking*: Strategier for å håndtere og forutse problemer i prosessen. Mulige strategier kan være å identifisere kilden til problemet, lese gjennom egen kode, eksperimentere med flere koder, prøve å kode det på nytt og finne eksempelkoder som fungerer.
- *Gjenbruk og remiksing*: Bygge på andres arbeider for hjelp til å finne ideer og koder å videreutvikle, som gir dem muligheten til å potensielt lage noe mer komplekst enn de kunne ha laget på egen hånd.
- *Abstrahering og modularisering*: Å bygge noe stort og komplekst ved å dele det i samlinger av mindre deler.

Perspektiver i algoritmisk tenkning

Til slutt la Brennan og Resnick (2012) til perspektivdimensjonen for å kunne beskrive endringer i elevers tankegang gjennom å jobbe med programmering og oppgaver med algoritmisk tenkning. Denne dimensjonen inneholder tre perspektiver.

- *Uttrykke*: En tankegang om at man kan skape, designe og være kreativ ved å tenke algoritmisk.
- *Koble sammen*: Man kan gjøre forskjellige ting og få mer ut av å jobbe sammen og diskutere, samt produsere eller utvide et program respektivt for og av andre.
- *Søkende*: Et tankesett om at man kan bruke programmering og algoritmisk tenkning for å stille spørsmål og ta stilling til muligheter og begrensninger ulike verktøy og hjelpemidler gir oss.

2.2.3 Udir sin definisjon av algoritmisk tenkning opp mot rammeverkene

Utdanningsdirektoratet (2019a) har i sin artikkel om algoritmisk tenkning en figur (Figur 2.3) over deres definisjon av CT med viktige nøkkelbegrep som inngår i algoritmisk tenkning og typiske arbeidsmåter den algoritmiske tenkeren bruker for å løse problemer. Denne artikkelen beskriver Utdanningsdirektoratets definisjon av algoritmisk tenkning som er den samme brukt i læreplanen. Dermed er det interessant å diskutere Figur 2.3 opp mot Brennan og Resnick (2012) sine nøkkeldimensjoner for algoritmisk tenkning, samt 5E-modellen for å se på likheter og forskjeller mellom rammeverkenes og Utdanningsdirektoratets definisjon av algoritmisk tenkning.

Figur 2.3. Utdanningsdirektoratets definisjon av algoritmisk tenkning med tilhørende nøkkelbegreper og arbeidsmåter relevant til ferdigheten. Hentet fra Utdanningsdirektoratet (2019).

I figuren kan vi gjenkjenne flere av Brennan og Resnick (2012) sine praksiser, som *testing og feilsøking* i arbeidsmåtene *fikle* og *feilsøke* eller *abstrahering og modularisering* i *dekomposisjon*, *abstraksjon* og *skape*, og deres perspektiver som for *eksempel koble sammen* i arbeidsmåten *samarbeide*. I tillegg kan vi finne noen konstrukter fra 5E-modellen som *Envisage [Forestille seg]* i *logikk*, *Explore [Utforske]* i *fikle* og *feilsøke*, og *Exchange [Dele]* i *samarbeide*, for å nevne noen. Det vi kan gjenkjenne fra Brennan og Resnick (2012) sin tredje dimensjon, *perspektiver i algoritmisk tenkning*, er at det innebærer den faktiske prosessen elevene må gjennom og har ikke fokus på selve oppgavene, som passer med valget om å se bort fra denne dimensjonen grunnet oppgavens art som lærebokanalyse, som blir beskrevet i Kapittel 2.2.4. Man ser at rammeverkene sammen samsvarer godt med Utdanningsdirektoratets tanker om hva som er essensielt med tanke på algoritmisk tenkning, og da også læreplanens definisjon av algoritmisk tenkning. Derfor kan det være passende å bruke rammeverkene til analysen av lærebøkene som har tatt utgangspunkt i Utdanningsdirektoratets retningslinjer og læreplanen.

Med tanke på definisjonen av algoritmisk tenkning ser vi at Utdanningsdirektoratets (2019) definisjon har et større fokus på den digitale kompetansen enn Wing (2006) som ønsket å få frem at algoritmisk tenkning ikke bare var en digital kompetanse. Algoritmisk tenkning er en ferdighet som krever tenkning på flere abstraksjonsnivåer og vil være en nyttig egenskap uansett hva slags yrke man tilhører. Dette står i tråd med det Bocconi et al. (2018) fant om at digital kompetanse var et hovedfokus i de nordiske landene i innføringen av algoritmisk tenkning inn i skolen. Samtidig er essensen den samme, der vi igjen kan kjenne igjen kjernekonseptene og arbeidsmåtene innenfor algoritmisk tenkning i Figur 2.3 (Bocconi et al., 2016; referert i Bocconi et al., 2018, s. 7).

2.2.4 Handlinger og begreper

Brennan og Resnick (2012) sitt CT-rammeverk mangler koblingen av programmering til den matematiske konteksten og er alene ikke passende for en lærebokanalyse. Bråting og Kilhamn (2021) opplevde at 5E rammeverket (Benton et al., 2016; Benton et al., 2017) ikke utdypet tilstrekkelig nok om begreper og praksiser innenfor algoritmisk tenkning. De fant at mange oppgaver i datamaterialet deres ikke inkluderte noen av prinsippene i 5E og kunne dermed ikke beskrive innholdet i lærebøkene omfattende nok for en innholdsanalyse. De valgte å kombinere 5E med Brennan og Resnick (2012) sitt rammeverk om algoritmisk tenkning, som ga dem et nytt rammeverk om *handlinger* og *begreper*. Bråting og Kilhamn (2021) valgte å se bort fra den tredje dimensjonen, *perspektiver i algoritmisk tenkning*, i Brennan og Resnick (2012) sitt rammeverk for algoritmisk tenkning. Denne oppgaven er en lærebokanalyse og det vil derfor være utfordrende å få innsikt i elevers tankegang, som gjør at denne dimensjonen vil være lite hensiktsmessig å ha med i rammeverket. Grunnet alt dette har jeg valgt å benytte meg av Bråting og Kilhamn (2021) sitt analytiske verktøy om handlinger og begreper der begrunnelsen for dette vil bli beskrevet i Kapittel 2.2.5.

Bråting og Kilhamn (2021) konstruerte et analytisk verktøy som består av *handlinger* og *begreper*. Dette verktøyet endte opp med seks handlinger elever kunne bli spurt om å gjøre i oppgaver, der man kan finne deler av de to originale rammeverkene. Man kan finne praksiser innenfor algoritmisk tenkning som *feilsøking* og å *være inkrementell* og *iterativ* fra Brennan og Resnick (2012), samt konstruktene *utforske*, *forklare* og *forestille seg* fra 5E-modellen. Bråting og Kilhamn (2021) fant at den første handlingen, *a) Følge en prosedyre*, dukket opp i deres data og valgte dermed å legge den inn i verktøyet. Under er en oversettelse av Bråting og Kilhamn (2021) sine handlinger og en kort beskrivelse av disse. Min tolkning og klassifisering av handlingene blir presentert i Kapittel 4.1.1.

a) *Følge en prosedyre* - følge instruksjoner stegvis, repetere eller fortsette et mønster.
b) *Finne regel* - finn ut prosedyren, regelen eller mønsteret som genererer et utfall, for eksempel en tallsekvens.
c) *Feilsøke* - feilsøk en kode.
d) Forme og skape - gi instruksjoner, lage mønstre, skrive kode, representer med symboler.
e) *Forklare* - forklare med naturlig språk, bruke ord for å beskrive en prosedyre, en regel, et mønster eller et konsept.
f) *Forestille seg* - forutse hva som vil skje, reflektere over mulige utfall når betingelser og verdier blir forandret.

Det analytiske verktøyet til Bråting og Kilhamn (2021) inkluderer to type begreper som er funnet i oppgavene: *matematiske begrep* og *programmeringsbegrep*. Begrepene blir identifisert og klassifisert ved hjelp av en deskriptiv analyse der man ser på ordbruk og meningen disse ordene får i kontekst av oppgaven og temaet som blir beskrevet i Kapittel 4.1.2. For eksempel kan *algoritme* bli klassifisert i begge begrepskategoriene og man må dermed se på hvordan oppgaven definerer ordet og hvordan algoritmen skal brukes. Til slutt vil man se på kombinasjonen av handlinger og begreper for å se om det knyttes, eller kunne potensielt knyttes, noen eksplisitte linker mellom programmering og matematikk og hvordan dette kan utvikle elevenes matematiske kunnskap (Bråting &

Kilhamn, 2021). Andre programmeringsbegreper funnet av Bråting og Kilhamn (2021) i svenske lærebøker var *stegvise instruksjoner*, *løkker*, *regel*, *kode*, *betingelse* og *feilsøking*. Jeg vil derfor også være på utkikk etter disse i analyseprosessen av de norske lærebøkene for å kunne kommentere forskjeller og likheter i de to landenes utførelse av å introdusere programmering inn i lærebøkene for matematikkfaget.

2.2.5 Hvorfor kan handlinger og begreper brukes

Algoritmisk tenkning og programmering er ansett som hjelpemidler som kan bidra til læringen av matematikk ved å øke motivasjonen hos misfornøyde elever (Bocconi et al., 2018). Det er dermed interessant å se om lærebøkene utnytter disse hjelpemidlene og hvordan de kan bidra til å utvikle elevenes matematiske kunnskaper ved hjelp av rammeverket til Bråting og Kilhamn (2021).

Med utviklingen av Scratch og innførelsen av algoritmisk tenkning, ble programmering til et nytt og eget fag i England kalt «Computing» (Bråting & Kilhamn, 2021). I Sverige har de derimot, som i Norge, integrert *digital kompetanse*, *programmering* og *algoritmisk tenkning* inn i læreplanen i matematikk, med noen små forskjeller i utførelse og definisjonen av algoritmisk tenkning (Bocconi et al., 2022). Disse ulikhetene vil gi min studie en annen vinkling enn Bråting og Kilhamn (2021) sin studie, som gjør det interessant å bruke det samme rammeverket, men fra det norske perspektivet. Ettersom jeg ville gjøre en liknende studie som Bråting og Kilhamn (2021) utførte i Sverige, samt at forskningsspørsmålene de har og mine er relativt like, virket det hensiktsmessig å ta utgangspunkt i rammeverket de utviklet til en innholdsanalyse av lærebøker.

Både Bråting og Kilhamn (2021) og Benton et al. (2017) har benyttet seg av 5E med Scratch som programmeringsverktøy. I norske lærebøker for 9. trinn har de derimot brukt programmeringsspråket Python som er tekstbasert istedenfor blokkbasert. Dette kan gi et annet fokuspunkt i oppgavene i norske lærebøker som baserer seg på tekstprogrammering. I tillegg skal jeg undersøke lærebøker for eldre elever enn det både Bråting og Kilhamn (2021) og Benton et al. (2016; 2017) brukte 5E-rammeverket på.

Som beskrevet i Kapittel 2.2.4 opplevde Bråting og Kilhamn (2021) at 5E-modellen og CT-rammeverket til Brennan og Resnick (2012) ikke dekket hele datamaterialet deres i studien, som førte til utviklingen av deres analyseverktøy. Av samme grunn anser jeg det som mer hensiktsmessig å ikke bruke 5E og Brennan og Resnick (2012) sine nøkkeldimensjoner direkte, men heller bruke Bråting og Kilhamn (2021) sine handlinger og begreper da det er flere likheter ved deres og min studie i forskningsspørsmålene og lærebøker som datamateriale.

Jeg anser Bråting og Kilhamn (2021) sitt analytiske verktøy som passende for å finne svar på mine forskningsspørsmål. Som nevnt, samsvarer deres forskningsspørsmål sterkt med mine egne, samt at innføringen av programmering og algoritmisk tenkning i skolen er gjort relativt likt i Sverige og Norge med plasseringen i allerede eksisterende fag. Det vil dermed være spennende å se om man vil finne de samme funnene i norske lærebøker som de svenske lærebøkene. I tillegg ser jeg på disse handlingene som like egnet for tekstprogrammering som blokkprogrammering, da de samme konseptene og konstruksjonene finnes i begge språktyper, men blir representert på ulike måter, og vil dermed være et hensiktsmessig analyseverktøy.

2.3 Sannsynlighet i læreplanen

Sannsynlighet er i LK20 nevnt i tre kompetansemål i læreplanen for grunnskolen: en gang for 5. trinn og to ganger for 9. trinn (Kunnskapsdepartementet, 2020a). Etter 5. trinn står det at eleven skal kunne «diskutere tilfeldighet og sannsynlighet i spill og praktiske situasjoner og knytte det til brøk», mens etter 9. trinn skal eleven kunne «beregne og vurdere sannsynlighet i statistikk og spill» og «simulere utfall i tilfeldige forsøk og beregne sannsynligheten for at noe skal inntreffe, ved å bruke programmering». Nytt i kompetansemålene for 9. trinn er introduksjonen av programmering som blir spesifikt nevnt i et av målene.

I tillegg til innføringen av programmering inn i matematikkfaget, ble kjerneelementer innført med den nye læreplanen. Kunnskapsløftet 2020 innførte kjerneelementer i alle fag som skal være det viktigste faglige innholdet elevene skal arbeide med i opplæringen (Utdanningsdirektoratet, 2019b). I matematikk består kjerneelementene av:

- utforskning og problemløsning
- modellering og anvendelser
- resonnering og argumentasjon
- representasjon og kommunikasjon
- abstraksjon og generalisering
- matematiske kunnskapsområder

Ifølge læreplanen for 9. trinn (Kunnskapsdepartementet, 2020a) støtter kjerneelementet *representasjon og kommunikasjon* kompetansemålet om å beregne og vurdere sannsynlighet i statistikk og spill, mens *utforskning og problemløsning* støtter det andre kompetansemålet som spesifikt nevner programmering. Under dette sistnevnte kjerneelementet blir algoritmisk tenkning nevnt som en viktig del av prosessen for å utvikle elevenes strategier og fremgangsmåter innenfor problemløsning, samt å vurdere nyttigheten av å bruke digitale verktøy (Kunnskapsdepartementet, 2020a). Fra læreplanen er det synlig at Kunnskapsdepartementet anser det som nødvendige og hensiktsmessige ferdigheter å tilegne seg i sannsynlighetstemaet og lagt programmeringen under dette temaet innenfor matematikk.

Bruker man *Utdanningsdirektoratets støtte til læreplanen* på nettsiden for læreplanen i matematikk for 9. trinn kommer det opp at kompetansemålet som inneholder både sannsynlighet og programmering på 9. trinn skal bygge på et kompetansemål fra 8. trinn som sier at elevene skal kunne «utforske hvordan algoritmer kan skapes, testes og forbedres ved hjelp av programmering» (Kunnskapsdepartementet, 2020a). Man ser dermed at eleven skal i prinsippet ha kjennskap til programmering og hvordan man lager, tester og modifiserer koder og algoritmer, men hvor mye kjennskap til og programmeringsferdigheter eleven skal ha er vanskelig å ta stilling til grunnet læreplanmålets uklarhet og bredde. Det første kompetansemålet om sannsynlighet fra 9. trinn blir ikke lenket med et tidligere kompetansemål. Ser man da på hva elevene skulle kunne etter 5. trinn, som var det eneste andre stedet sannsynlighet ble nevnt i kompetansemålene, handlet det om å diskutere sjanse og sannsynlighet i spill og praktiske situasjoner ved hjelp av brøk, som kan ses på som en forgjenger og introduksjon til å jobbe med det første kompetansemålet fra 9. trinn om å beregne og vurdere sannsynlighet i statistikk og spill. Man kan dermed si at en standard elev, når de kommer til sannsynlighetstemaet i 9. trinn, skal ha vært introdusert for sannsynlighet og opparbeidet seg en viss intuisjon og forforståelse for enkle beregninger for hånd, samt ha nok programmeringskunnskap til å kunne skrive egen kode selv.

Grunnet studiens rot i lærebøkene, og mangel på elevmedvirkning er det utfordrende å vite hvordan elevene vil oppleve og løse oppgavene jeg har analysert. I tillegg har hver elev ulik kunnskap, ferdigheter og holdninger til matematikk og programmering, som jeg må se bort fra. Jeg har dermed formulert hva en standard elev i teorien vil ha av kunnskap før man begynner på sannsynlighetstemaet på 9. trinn ut fra hva Utdanningsdirektoratet forventer at en elev skal lære seg ved bruk av læreplanmålene. Dette gir en basis for hva jeg kan forvente av elevene og en standard å bruke for analysen av lærebøkene.

2.4 Læring av sannsynlighet

Bråting og Kilhamn (2021) sitt analyseverktøy vil hjelpe meg å kartlegge datamaterialet og se på hvordan programmeringen blir benyttet og hvordan dette kobles til matematikken, men det sier ikke noe om innholdet er gunstig for læring av sannsynlighet. For dette trenger man en didaktisk modell om læring av sannsynlighet. Fra litteraturen fant jeg rammeverkene til Castro (1998) og Wilensky (1995) som har utviklet ulike prinsipper de mener er viktig for å gi elever en større forståelse og kunnskap innenfor sannsynlighet. Videre vil jeg presentere Castro (1998) sine tre krav og Wilensky (1995) sine nøkkelelementer og min syntese av disse prinsippene, samt en begrunnelse for hvorfor min didaktiske modell passet til oppgaven.

2.4.1 Castro sine tre krav

Tradisjonelt sett har sannsynlighetsundervisning fulgt den logiske strukturen av disiplinen lineært, uten å ta i betraktning den kognitive læringsprosessen (Castro, 1998). Den tradisjonelle undervisningsmetoden har derimot vist seg å representere en hindring i klasserommet og læringen av sannsynlighet (Batanero et al., *The Nature of Chance and Probability*, 2005; referert i Serpe & Frassia, 2017). Didaktikere har dermed prøvd å introdusere nye verktøy for å øke elevenes forståelse, men er ofte mislykkede da det er for små justeringer gjort til den tradisjonelle undervisningen og blir innført mer som sideprosjekter enn en del av matematikken (Castro, 1998). Man mangler det å gjøre matematiske sammenhenger eksplisitt, noe Castro (1998) prøver å overkomme med sitt didaktiske forslag. Castro (1998) definerer sin undervisningsmodell ut fra tre kriterier:

1. Behovet for å ta hensyn til forforståelsen og intuisjonen hos elever angående sjanse og sannsynlighet.

Det er ikke nok å undervise i henhold til den logiske strukturen til sannsynlighetsteori, vi må ta hensyn til den psykologiske strukturen til temaet når elevene konfronterer tilfeldige hendelser og kan oppleve strid med deres intuisjon (Castro, 1998).

2. Behovet for å gi et empirisk fokus til læren om sannsynlighetsteori.

Siden spill, sjanse og tilfeldige eksperimenter var av stor betydning i den historiske utviklingen av begreper og lover innenfor sannsynlighet, må det fortsette å ha en viktig rolle i dagens undervisning av temaet (Castro, 1998). Dette empiriske fokuset er det som kan provosere frem en nødvendig kognitiv konflikt hos elevene for å endre deres kunnskap og intuisjon innenfor sannsynlighet.

3. Behovet for å gi spesifikk opplæring i sannsynlighetsresonnement.

Ren undervisning av begreper og lover i sannsynlighetsregning vil ikke garantere å minske misoppfatninger i elevenes resonnement. Sammenhengen mellom intuisjon og matematikk er basisen for å få til en pedagogisk intervensjon (Fischbein, 1987; Poincaré, 1952; referert i Castro, 1998).

2.4.2 Wilensky sine nøkkelelementer i Connected Mathematics

Sannsynlighetstemaet blir ofte sett på som en samling av formler som må pugges av elevene (Wilensky, 1995). Svært lite blir gjort i skolen for å utforske og prøve å finne svar på grunnleggende ideer innenfor sannsynlighet, som «hvordan kan noe være både tilfeldig og strukturert?», da det blir antatt at det vil være for utfordrende for elever å ta stilling til. «Trygg sannsynlighet», som Wilensky (1995) kaller det, er best undervist gjennom formelle oppgaver uten å gi for mye oppmerksomhet til underliggende konsepter. For eksempel forteller Wilensky (1995) at siden det er en vanlig konklusjon at man ikke kan stole på sin intuisjon, så skal man i undervisningen innstille elevene med en mistillit til deres intuitive tanker og respekt for formlene. Dessverre kan dette føre til at selv om elevene pugger formlene og lærer i hvilke kontekster de er hensiktsmessig å benytte seg av, er det ikke sikkert at elevene egentlig vet hva man driver med og hvorfor. Elever kan dermed miste muligheten til å få tilgang til fundamentale sannsynlighetsforestillinger som er kraftige måter å gjøre seg en mening om verden og hvorfor man skal bry seg om å lære om sannsynlighet (Wilensky, 1995).

Connected Mathematics har grunnlag i ideen om at matematiske konsepter ikke bare har én mening, og det er bare gjennom å koble matematiske konsepter at man får en større forståelse for sammenhengen i matematikk (Wilensky, 1995). Rammeverket ble utviklet for å fremme en dypere og mer sammenkoblet forståelse av matematikk ved å gi elever muligheter til å observere og utforske i et virtuelt, dynamisk læringsmiljø. Nøkkelelementene innenfor Connected Mathematics er som følger:

1) Utforske flere betydninger av begreper og lage sammenhenger mellom de forskjellige representasjonene.
2) Et fokus på epistemologiske problemer der eleven, ved å stille spørsmål, kan utvikle en større forståelse innenfor sannsynlighet og for verden.
3) Bruken av paradoks, det vil si erkjenne at det er strid mellom to konsepter som må binde sammen og som dermed genererer ny matematikk.
4) Gjennomføre en eleveid undersøkelse som den sentrale aktiviteten i matematisk læring.
5) Anerkjennelse av og oppmerksomhet til elevens møte med frustrasjon og skam i møte med konflikter i matematikk.
6) Lage matematikk (og uttrykke det på en konkret form).
7) Bruken av teknologi som et medium for å lage og uttrykke matematikk.

Wilensky (1995) fant at ved hjelp av disse nøkkelelementene kan elever lage dypere sannsynlighetsargumenter som undersøker fundamentene i sannsynlighet. Ved å ha fått en dypere forståelse for fundamentale konsepter på denne måten, utviklet det seg en sterk intuitiv forståelse av konsepter som tilfeldighet, distribusjon og forventning. Solid intuisjon rundt sannsynlighet ble klart utviklet hos elever i denne studien.

Tilgjengeligheten til programmering, og det miljøet verktøyet introduserte, tilrettela for oppnåelsen av mange av disse målene.

2.4.3 Syntese av Castro og Wilensky sine rammeverk

Jeg har laget en syntese ut fra Castro (1998) sine krav og Wilensky (1995) sine nøkkelelementer. Ved å se på hva som er felles i rammeverkene, utviklet jeg en modell som tar for seg de viktigste aspektene fra Castro (1998) og Wilensky (1995). Dette gir meg en mer praktisk modell å benytte under analysen, og vil trekke frem de viktigste elementene for undervisning av sannsynlighet ut fra rammeverkene. Jeg har da sett over alle punktene og satt de sammen til fire kategorier:

1. Elevenes tankegang og intuisjon
2. Knytte virkelighet og teori sammen
3. Matematiske sammenhenger
4. Oppsett

Videre vil jeg utdype om disse kategoriene og hvilke krav og elementer de består av, samt hva som måtte kuttes bort. Alle kategoriene har to spørsmål som er til hjelp for hva man skal se etter i analysen av datamaterialet.

1. Elevenes tankegang og intuisjon: *Blir elevenes intuisjon og forforståelse utfordret i oppgaven? Blir det tatt opp misoppfatninger innenfor sannsynlighet?*

Dette er et krav som går mer på intuisjon og forforståelse, og å minske misoppfatninger. Dette vil være hypotetisk da man bare kan se på oppgavene og tenke over hva målet er med dem. Jeg vil derfor gå ut fra kunnskapen og forforståelsen standardeleven formulert i Kapittel 2.2 vil ha i analysen av lærebøkene. Denne kategorien er satt sammen av Castro (1998) sitt første og tredje krav.

2. Knytte virkelighet og teori sammen: *Har oppgavene et empirisk fokus eller er virkelighetsnære? Blir det stilt spørsmål til elevene som kan utvikle deres sannsynlighetskunnskap ved hjelp av empiri/virkeligheten?*

I dette kravet er settingen og empiri i fokus, og om oppgaven har som hensikt å øke forståelsen hos elevene innenfor sannsynlighetstemaet. Kravet er satt sammen av Castro (1998) sitt andre krav og det andre nøkkelelementet til Wilensky (1995).

3. Matematiske sammenhenger: *Bygges det forbindelser mellom forskjellige sannsynlighetsbegreper i oppgavene? Blir forskjellige begreper satt opp mot hverandre og stilt spørsmål ved for å utvikle elevenes kunnskap videre?*

Her vil begreper og konsepter være i fokus og hvordan oppgavene får frem sammenhenger og logiske brister mellom dem. Dette kravet består av Wilensky (1995) sine første og tredje nøkkelelementer.

4. Oppsett: *Hvordan skal oppgaven utføres? Hva slags verktøy skal benyttes?*

I Wilensky (1995) sine fjerde, sjette og syvende nøkkelelementer er fokuset mer på oppsettet til oppgaven og hva slags verktøy elevene kan bruke. Jeg valgte dermed å sette disse sammen til den fjerde kategorien om oppsett i oppgaven.

Til slutt har jeg valgt å ta bort det femte nøkkelelementet til Wilensky (1995). Dette elementet tar for seg at man skal anerkjenne elevenes frustrasjon og skam i møte med konflikter av matematikk, som er vanskelig å finne ut av i en lærebokanalyse uten observasjon eller intervju av elever.

2.4.4 Begrunnelse av Castro, Wilensky og utviklede kategorier

Til nå har det vært et fokus på hvordan programmering kommer frem i oppgavene og om det er koblet med matematikken, men ikke så mye om elevene vil få hensiktsmessig sannsynlighetskunnskap ut fra oppgavene. Med disse fire kategoriene fra syntesen av Castro (1998) og Wilensky (1995) sine prinsipper håper jeg at det er mulig å kunne hypotetisk diskutere om det finnes oppgaver i lærebøkene som kan fremme forståelse innenfor sannsynlighet og hvordan de går frem for å få det til.

Et fokus i kategoriene er å kunne se sammenhenger og knytte matematikken med empiri og virkeligheten. Utdanningsdirektoratet anser dette som sentralt da det eksplisitt blir nevnt om fagets relevans og sentrale verdier: «Matematikk er et sentralt fag for å kunne forstå mønstre og sammenhenger i samfunnet og naturen gjennom modellering og anvendelser» (Kunnskapsdepartementet, 2020a). Her kommer i tillegg kategorien om oppsett og verktøy inn med modellering. Under fagets relevans og sentrale verdier blir det også skrevet at «kritisk tenkning i matematikk omfatter kritisk vurdering av resonnementer og argumenter og kan ruste elevene til å gjøre egne valg og ta stilling til viktige spørsmål i sitt eget liv og i samfunnet», som kan samsvare med kategorien om elevenes tankegang og intuisjon og hvorfor dette er viktig å utvikle hos eleven.

I tillegg kan man kjenne igjen flere momenter fra kjerneelementene i kategoriene, som at det er mer vekt på strategi og fremgangsmåte enn løsning (*utforskning og problemløsning*), hvordan matematikk kan brukes både i og utenfor faget samt innsikt i matematisk modeller (*modellering og anvendelser*). Det står også at elevene skal lære at matematiske regler og resultater ikke er tilfeldig og har klare begrunnelser og da utvikle deres forståelse og intuisjon innenfor matematikk (*resonnering og argumentasjon*), og oversette og veksle mellom matematiske representasjoner og forstå sammenhengen mellom dem (*representasjon og kommunikasjon*). Til slutt har vi at elevene skal utvikle sin tankegang og oppdage sammenhenger og strukturer (*abstraksjon og generalisering*), samt at det står spesifikt i kjerneelementet *matematiske kunnskapsområder* at «kunnskap om sannsynlighet gir elevene et godt grunnlag når de skal gjøre valg i sitt eget liv, i samfunnet og i arbeidslivet», som viser viktigheten av å utvikle elevenes kunnskap innenfor sannsynlighet (Kunnskapsdepartementet, 2020a).

2.5 Programmering i sannsynlighet

Gjennom forskning har det vist seg at i sannsynlighetsundervisning er den tradisjonelle undervisningsmetoden sett på som en hindring i flere land (Batanero et al., *The Nature of Chance and Probability*, 2005; referert i Serpe & Frassia, 2017). Sannsynlighet er utfordrende å undervise av ulike årsaker, inkludert et misforhold mellom intuisjon og begrepsutvikling selv når det gjelder tilsynelatende elementære begreper (Batanero et al., 2005). Serpe og Frassia (2017) fant at ved å bruke simulering i undervisningen av sannsynlighet kan man overkomme mange av disse hindringene. Når man underviser sannsynlighet er det viktig å lage effektive representasjoner som kan forklare hvorfor noen metoder funker til visse kontekster og gir bistand til generalisering. Ved hjelp av programmering kan man oppnå simulering i skolekontekst, der elever kan modellere og utforske tilfeldige fenomener og predikere langsiktig oppførsel og få et nytt perspektiv og utvikle en større forståelse innenfor sannsynlighet (Batanero et al., 2005). Å undervise i sannsynlighet er utfordrende, da det krever mer enn bare å presentere ulike modeller og deres anvendelser. Det involverer en dypere utforskning av bredere spørsmål, inkludert:

- hvordan man henter kunnskap fra data
- begrunnelser for modellens egnethet
- hvordan man veileder elevene til å utvikle riktig intuisjon innenfor sannsynlighet

Her kan programmering være en viktig ressurs i undervisningen som et nyttig verktøy for å representere matematiske objekter ved hjelp av simulering. Ved å anvende algoritmer og implementering av simulasjoner kan det hjelpe elevene til å forstå viktigheten av å bruke modeller for å beskrive virkeligheten, og samtidig formulere klart og detaljert teorien som ligger til grunn for fenomenet som skal representeres (Serpe & Frassia, 2017). Simulering ved hjelp av programmering spiller også en vesentlig rolle i å utvikle elevenes problemløsningsferdigheter. Bruk av programmering kan hjelpe elever å utforske ulike metoder for å finne en løsning og trene deres evne til å «forutse» og «klare å se» i matematikk, samt å bryte ned store problemer til flere enkle som er mulige å løse. Samtidig poengterer Batanero et al. (2005) at siden simulering bare gir problemløsningen og ikke grunnen til at løsningen er gyldig, har den ingen forklaringskraft og burde støttes opp av den mer tradisjonelle undervisningen av de formelle reglene innenfor sannsynlighet.

Biehler (2019) skiller mellom to pedagogiske funksjoner ved simulering. Den første funksjonen er simulering for visualisering og for å gjøre sannsynlighet og sjanse/tilfeldighet til en del av elevenes opplevelse. Den andre går ut på simulering som en del av prosessen med modellering og problemløsning. Denne siste pedagogiske funksjonen krever at elevene oppnår tilstrekkelig kompetanse i å bruke det digitale verktøyet slik at de kan konstruere egne modeller og simuleringer (Biehler, 2019). Dermed utfører ikke elevene bare simuleringer laget på forhånd eller observerer simuleringer gjort av læreren.

Det finnes dermed en nytteverdi i å utføre flere simuleringer i sannsynlighetsundervisning, ved bruk av digitale verktøy og programmering. Dette kan øke elevenes forståelse innenfor temaet og gi dem kjennskap til ulike representasjoner for ulike problemer og hva slags modeller som er hensiktsmessig å bruke. Simulering ved bruk av programmering må samtidig få en plass i den helhetlige undervisningen av sannsynlighet og knyttes til undervisning og problemer uten bruk av digitale hjelpemidler for økt dybdelæring hos elevene.

3 Metode

I dette kapittelet vil jeg presentere metodologien som ligger til grunn for å svare på forskningsspørsmålene: *(1) Hva karakteriserer programmeringsinnholdet i sannsynlighetstemaet i norske matematikkbøker for 9. trinn?* og *(2) På hvilke måter knytter lærebøkene programmering og sannsynlighet sammen?* I Kapittel 3.1 beskrives innholdsanalyse som er den valgte metoden i denne studien, før jeg i Kapittel 3.2 forteller om utvalget jeg endte opp med ut fra forskningsspørsmålene. Til slutt vil jeg i Kapittel 3.3 og 3.4 vurdere studiens reliabilitet og validitet og gjøre rede for etiske betraktninger respektivt.

3.1 Forskningsmetode

Studier som omhandler en analyse av innholdet i lærebøker, er som oftest gjennomført ved bruk av innholdsanalyse. Innholdsanalyse er en forskningsmetode for å gjennomføre replikerbare og gyldige konklusjoner fra tekster til konteksten de er laget for (Krippendorff, 2013, s. 19). Med denne forskningsmetoden kan man dermed svare på spørsmål om innholdet i lærebøkene og relasjonen mellom innholdet i kontekst til hvordan de skal brukes (Rezat & Strässer, 2017, s. 508). En innholdsanalyse vil derfor være en passende forskningsmetode for å undersøke programmeringsinnholdet i matematikklærebøkene for 9. trinn og få et innblikk i hva elevene kan få ut av eksempler og oppgaver som inneholder programmering. Med tanke på at programmering og CT er nye konsepter i matematikkfaget, og dermed i lærebøkene, er det passende at det overveiende målet med innholdsanalyse er å trekke slutninger fra innholdet til dens innvirkning på elever og lærere (Rezat & Strässer, 2017, s. 500). Dette kan gi en pekepinn på hva lærebøkene forventer av programmeringskunnskap og læring hos elevene.

Kvalitativ innholdsanalyse og dens metode kjennetegnes av fokuset på kategoriene datamaterialet kodes med (Kuckartz, 2019, s. 195). Kategoriene kan kodes på to ulike måter: deduktivt og induktivt. I en deduktiv koding benytter man seg av kategorier fra et teoretisk rammeverk og koder datamaterialet etter disse allerede eksisterende kategoriene. Man kan også jobbe induktivt ved at kategoriene oppstår fra analysen av datamaterialet, der man plukker ut med et så åpent sinn som mulig og jobber seg mot en teori. For begge metoder er det viktig å beskrive prosessen så presist som mulig for å sikre at kodeprosessen er pålitelig. I min studie endte jeg opp med både deduktiv og induktiv analyse av datamaterialet, grunnet forskjeller mellom min og Bråting og Kilhamn (2021) sin studie som førte til utvidelser og tilpasninger av rammeverket om handlinger og begreper. Innenfor programmeringsbegreper ble rammeverket utvidet med to nye begreper som ble hyppig brukt i de norske lærebøkene. I tillegg valgte jeg å kombinere to programmeringsbegrep, *algoritme* og *stegvise instruksjoner*, fra de originale begrepene fra Bråting og Kilhamn (2021) til et begrep, da jeg opplevde de hadde samme betydning i de norske lærebøkene. For de matematiske begrepene innså jeg fort at jeg ikke kunne ta utgangspunkt i begrepene til Bråting og Kilhamn (2021) da de har sett på alle temaene som inneholder programmering i lærebøkene for barnetrinnet, mens jeg spesifikt ser på sannsynlighetstemaet i lærebøkene for 9. trinn. Jeg valgte derfor i

starten å jobbe induktivt og identifiserte alle begreper som hadde en matematisk mening i kontekst av analyseenheten, før jeg plasserte disse i passende kategorier. Etter å ha utarbeidet kategoriene for matematiske begreper for min studie gikk jeg tilbake til datamaterialet og kodet ut fra disse.

All data som er relevant for forskningsspørsmålene må kodes fullstendig. Kodeprosessen foregikk i flere sykluser der jeg stadig sammenliknet og kontrasterte dataene opp mot hverandre for å sikre at jeg kodet datamaterialet til hensiktsmessige kategorier. Man kan analysere datamaterialet både kvalitativt og kvantitativt. Jeg har utført en kvalitativ analyse, der jeg har identifisert forskjeller og likheter i forskjellige lærebøker fra datamaterialet fra resultatene av analyseprosessen og sammenliknet funnene opp mot hverandre og relevant teori. Analysen ble delt i tre deler, der to av delene baserer seg på Bråting og Kilhamn (2021) sitt rammeverk med handlinger og begreper, mens den siste delanalysen går ut fra min utviklede modell om læring av sannsynlighet beskrevet i Kapittel 2.4. En full beskrivelse av analyseprosessen med mine valg og tolkninger av rammeverkene blir presentert i Kapittel 4.1.

3.2 Utvalg

Et utvalg er i forskning brukt som en beskrivelse av en del av en populasjon (Grønmo, 2023). Begrepet populasjon betegner ikke kun mennesker, men kan også referere til tekster (Cohen et al., 2018, s. 676). I min studie har jeg gjort et strategisk utvalg som har bygget på hvilke enheter som har vært mest relevant og interessant å inkludere ut fra mine forskningsspørsmål (Grønmo, 2023). Utvalget mitt har bestått av lærebøker for matematikkfaget fra ulike forlag. Videre vil jeg fortelle om avgrensningen av utvalget mitt og definere analyseenheter i datamaterialet.

3.2.1 Avgrensning av datamaterialet

På grunn av forskningsspørsmålene mine har det relevante utvalget av data blitt begrenset til lærebøker for matematikk på 9. trinn. I tillegg er dette utvalget innskrenket til sannsynlighetskapitlene i lærebøkene da jeg fokuserer på programmeringsinnholdet i dette temaet. Siden jeg ikke undersøker programmeringsinnholdet i flere temaer så blir datamaterialet begrenset, og derfor kan jeg ikke si noe om representasjonen og bruken av programmering i andre matematiske grener enn sannsynlighet. Men, ved å kun undersøke et tema kan jeg få en dypere forståelse for programmering sin plass og utnyttelse for spesifikt sannsynlighet som jeg ikke kunne fått ved å undersøke flere temaer. Programmering er nytt i matematikkfaget med LK20 som gjør at jeg kun kan undersøke lærebøker utgitt etter innføringen av den nye læreplanen.

Selv om utvalget var begrenset til trinn og tema, måtte jeg ta stilling til hvilke lærebøker som skulle være datagrunnlaget for studien. Jeg tar bare utgangspunkt i de fysiske lærebøkene da skolene i første omgang kjøper inn fysiske bøker når de trenger nye læreverk, der de ulike ressursene forlagene tilbyr nødvendigvis ikke vil være tilgjengelig for elevene med en gang etter byttet av lærebøker. Andre ressurser enn de fysiske lærebøkene vil dermed ikke være en del av datamaterialet i min studie. Jeg startet med å få en oversikt over de største forlagene i Norge som tilbyr lærebøker i matematikk, siden dette kan gi en indikasjon på hvilke lærebøker skolene i Norge kjøper inn (Neraal, 2024). De tre største forlagene har hver en lærebok for matematikk på 9. trinn:

Maximum 9 fra Gyldendal Norsk Forlag (Tofteberg et al., 2021), Matematikk 9 fra Cappelen Damm (Hjardar & Pedersen, 2021) og Matemagisk 9 fra Aschehoug (Kongsnes & Wallace, 2022). Videre i teksten vil jeg respektivt henvise til de ulike lærebøkene som Maximum, Matematikk 9 og Matemagisk. Grunnet utfordringer ved å få tilgang til lærebøkene endte jeg opp med Matemagisk og Maximum som mitt datamateriale da jeg kunne låne disse fra praksisskolen min. Matematikk 9 ble dermed valgt vekk fra utvalget. Matemagisk og Maximum er to læreverk mye brukt i norsk skole og det vil dermed være interessant å se på disse, samt at to lærebøker allerede gir et fyldig nok datamaterialet og det er dermed ikke nødvendig å undersøke tre lærebøker. Etter diskusjon med veileder anså vi det som et passende utvalg i størrelse ved å benytte seg av sannsynlighetskapitlene i Maximum og Matemagisk. Fra dataen ble det fort synlig at Maximum og Matemagisk har ulike tilnærminger i oppbygning og oppgavetyper i sannsynlighetstemaet, som gjør det mulig å undersøke ulike løsninger på innføringen av programmering i sannsynlighet.

3.2.2 Analyseenheter

Cohen et al. (2018) definerer en analyseenhet som alt fra et ord til et avsnitt, en hel tekst eller et tema. I denne studien er en analyseenhet definert som en oppgave eller et eksempel. Med oppgave mener jeg alt i det utvalgte datamaterialet som er enten nummerert med et oppgavenummer eller i en egen rute, som stiller et spørsmål eller gir et gjøremål til leseren. Om en oppgave inneholder deloppgaver, figurer eller eksempler som er relevante for oppgaven, vil disse tilhøre samme analyseenhet og ikke splittes opp i ulike enheter.

Både Matemagisk og Maximum inneholdt nummererte oppgaver, i tillegg til andre oppgavetyper. Matemagisk har oppgaver kalt *snakke matte*, *topptur* og *ekspedisjon*. *Snakke matte* er oppgaver elevene kan løse uten skrivebok og kjennetegnes av en blå rute rundt oppgaven. *Topptur* og *ekspedisjon* er omfattende oppgaver som kan anses som mer utfordrende i innhold og forventning til elevenes kunnskap. I Maximum finnes det oppgaver kalt *oppdrag* og *utforsk*, som også er mer omfattende oppgaver, men er ment som enten en introduksjon eller å utforske et fenomen.

I Matemagisk er eksemplene ikke relevant for studien da de kun var uten programmeringsinnhold. I Maximum finnes det et eksempel om simulering i Python og et eksempel med løsningsforslag av et av målene for kapitlene, som er tatt med i datamaterialet da det inneholder programmering.

3.3 Troverdighet

Innenfor kvalitativ forskning benyttes ofte de tre kriteriene: *pålitelighet*, *gyldighet* og *generaliserbarhet* som indikatorer på kvaliteten av en studie, samt to ekstra kriterier om *transparens* og *refleksivitet* (Tjora, 2017, s. 231). Videre vil jeg presentere disse fem kriteriene og hvordan jeg har forholdt meg til disse for å øke troverdigheten til min studie.

Gyldighet knyttes til den logiske sammenhengen mellom studiens utforming og funn, og om hvorvidt de funnene jeg får i studien faktisk er et svar på forskningsspørsmålene jeg stiller (Tjora, 2017, s. 232). Gyldigheten til en studie kan styrkes ved å tydeliggjøre

hvordan studien har blitt gjennomført ut fra forskningsspørsmålene. Gjennom å beskrive og begrunne de teoretiske rammeverkene brukt i Kapittel 2, metode i Kapittel 3 og analyseprosessen i Kapittel 4 har jeg prøvd å gjøre det eksplisitt hvordan og hvorfor jeg har gjort ulike valg der forskningsspørsmålene ligger i grunn.

Kriteriet om *pålitelighet* handler om sammenhengen i studien og forskerens posisjon. Med sammenheng i studien menes det om de teoretiske rammeverkene brukt er hensiktsmessige for forskningsspørsmålene, at valgt metode og analyseprosess vil igjen være passende og at funnene gjort blir diskutert opp mot relevant teori. For en styrket pålitelighet bør man dermed redegjøre for alle valg man tar, som hvordan datamaterialet er valgt ut i Kapittel 3.2 eller hvordan jeg har valgt ut analyseenheter som eksempler i Kapittel 4.1.4, der jeg gir en forklaring på disse valgene. Hva slags relasjoner det er mellom forskeren og utvalget kan også ha betydning for påliteligheten (Tjora, 2017, s. 237). Jeg har ikke benyttet meg av disse bøkene i tidligere skolegang og kun vært borti andre temaer enn sannsynlighet i praksis og vikartimer. Derfor har jeg lite tilknytning til både Matemagisk og Maximum og hvordan programmeringsinnholdet i datamaterialet er utvalgt. Programmering var ikke et tema da jeg var elev, som betyr at jeg heller ikke har noe forkunnskap om undervisning i emnet fra en elev sitt ståsted. Jeg hadde lite kunnskap og formeninger om hvordan de ulike forlagene gikk frem i sine læreverk før studien, og jeg var dermed en nøytral part som har analysert bøkene ut fra de teoretiske rammeverkene beskrevet i Kapittel 2.

Med *generalisering* stilles det spørsmål til forskningens relevans utover de enheter som faktisk er undersøkt (Tjora, 2017, s. 231). Det er ikke alle studier som har som mål å generalisere, som i min studie der jeg har satt søkelys på et spesifikt område i matematikken og hvordan programmeringsinnholdet er der. Ut fra mine funn kan jeg kunne si noe generelt i forhold til bruk av programmering i sannsynlighetstemaet. Det skal tas i betraktning at jeg kun har undersøkt to lærebøker, men siden de har ulik tilnærming til temaet kan det gi et visst overblikk for ulike løsninger av programmering i sannsynlighet. En form for generalisering i kvalitativ forskning er *naturalistisk generalisering*, som kan gjøres mulig ved å beskrive forskningen så detaljert som mulig slik at leseren kan vurdere hvorvidt funnene vil ha gyldighet for eksempel for leserens egen forskning (Tjora, 2017, s. 239). Dette kan gi en forskningsmessig nytte da andre forskere i sin lesning av min studie kan vurdere dens gyldighet. Men man kan ikke forvente at alle lesere har så god informasjon om alle forutsetninger for en studie at vedkommende skal kunne gjøre vurderingen av generaliserbarhet (Tjora, 2017, s. 240). Det at naturalistisk generalisering overlates til leseren kan være problematisk så lenge generalisering var et av målene ved forskningen. Generalisering har ikke vært et av mine mål ved studien, men jeg har likevel prøvd å beskrive bakgrunn, teori, prosess og resultater så detaljert som mulig for at leseren kan vurdere studiens reliabilitet og validitet.

Mens pålitelighet og gyldighet reflekterer om valgene gjort i studien har vært gode, handler *transparens* om hvor godt disse valgene formidles i rapporteringen av forskningen (Tjora, 2017, s. 248). Målet er at lesere har fått et så godt innblikk i forskningen, valgene gjort, og ulike tolkninger at de kan ta stilling til forskningens kvalitet. Om valg av tema er motivert av noe underliggende må også komme frem, noe jeg har beskrevet i Kapittel 1 for å sikre at jeg ikke har noe ytre eller indre påvirkning som kan føre til en interesse for bestemte funn. Det er vanlig i kvalitativ forskning å bruke utdrag fra datamaterialet, som jeg har gjort i Kapittel 4, for å gi leseren en mulighet til å møte empirien uten forskerens tolkning. Jeg kan ikke legge ved alle

analyseenheter fra datamaterialet, men et begrenset antall vil være mulig å vise frem uten at opphavsretten til lærebøkene blir brutt.

Til slutt har vi *refleksivitet*, som handler om å undersøke ens egen tolkning i studien og hva som påvirker den (Tjora, 2017, s. 251). Ved å ikke utføre analysen i et totalt sosialt vakuum kan man gjøre forskningen mer refleksiv, som jeg har gjort ved å diskutere mine tolkninger, usikkerheter og uklarheter med veileder og en annen medstudent som har kjennskap til rammeverkene benyttet i studien.

3.4 Etiske betraktninger

Grunnet studiens valg av innholdsanalyse som metode vil det være færre etiske begrensninger enn i studier med elever eller andre informanter, siden man kan i en slik analyse generere empiriske data uten at ikke-forskende deltakere er involvert (Tjora, 2017, s. 24). Det er derfor ikke nødvendig å melde prosjektet til Sikt - kunnskapssektorens tjenesteleverandør.

Det er likevel viktig å vurdere og ta hensyn til ulike virkninger forskningen kan ha på en ikke-deltakende tredjepart for å sikre deres personvern og rettigheter (NESH, 2021). I min studie kan forfatterne av lærebøkene være en slik tredjepart og jeg har derfor tenkt over hvordan jeg ordlegger meg når jeg skriver om lærebøkene og har analysert så objektivt som mulig ut fra rammeverkene der mine vurderinger i løpet av studien ikke er ment som en bedømmelse av kvaliteten av bøkene. Men, siden det er menneskelig innblanding i analysen, og alle mennesker er forskjellig, kan det føre til ulik tolkning av samme materiale. Min bakgrunn og mine formeninger vil dermed kunne påvirke, men det er da viktig å beskrive dette i sin helhet og prøve å være en nøytral forsker og bygge tolkningen på forskningsbaserte teorier, begreper og perspektiv (NESH, 2021). Det er i tillegg viktig å referere til hvor utvalget er fra, som er lærebøkene Matemagisk og Maximum, da datamaterialet ikke er mitt eget.

4 Analyse og resultater

I denne studien er forskningsspørsmålene: *(1) Hva karakteriserer programmeringsinnholdet i sannsynlighetstemaet i norske matematikkbøker for 9. trinn?* og *(2) På hvilke måter knytter lærebøkene programmering og sannsynlighet sammen?* For å svare på dette har jeg tatt utgangspunkt i analyseverktøyet utviklet av Bråting og Kilhamn (2021), samt de utviklede kategoriene basert på Castro (1998) og Wilensky (1995) sine rammeverk innenfor sannsynlighet. Det har blitt utført tre delanalyser: *handlinger*, *begreper* og *læring av sannsynlighet*. Delanalysene handlinger og begreper tar utgangspunkt i Bråting og Kilhamn (2021) sitt rammeverk, mens delanalysen om læring av sannsynlighet baserer seg på kategoriene laget fra Castro (1998) og Wilensky (1995) sine rammeverk. Disse delanalysene vil videre beskrives før det presenteres funn fra hver delanalyse individuelt.

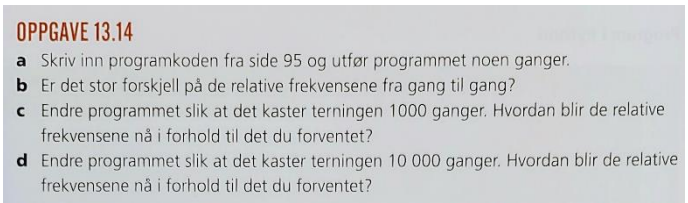
4.1 Analyseprosessen

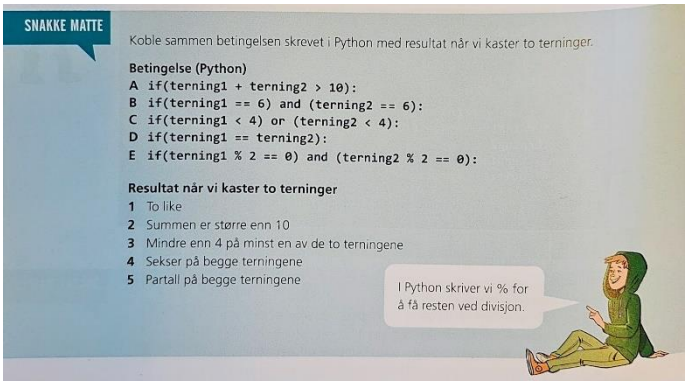
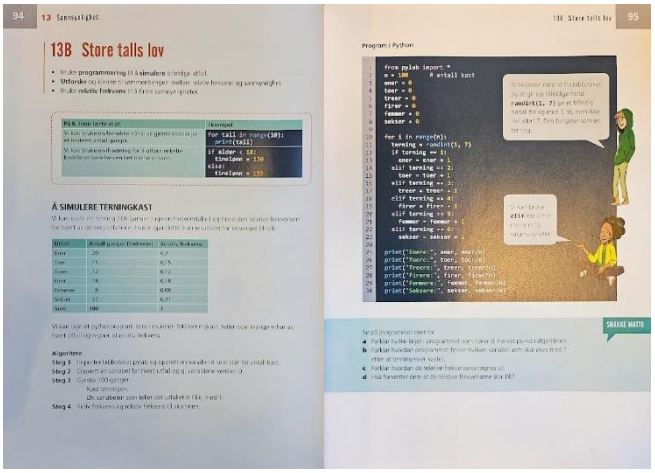
Til sammen har 30 analyseenheter, 16 fra Matemagisk og 14 fra Maximum, blitt analysert. Analyseenhetene ble analysert ved bruk av alle tre delanalyser samtidig. Under analysen ble beskrivelsen av standardeleven i Kapittel 2.3 benyttet som et utgangspunkt for forkunnskap og ferdigheter en elev kan ha innenfor programmering og matematikk før de begynner med sannsynlighet på 9. trinn.

4.1.1 Delanalyse 1: Handlinger

I analysen av handlinger ble det brukt koder utviklet av Bråting og Kilhamn (2021), som beskrevet i Kapittel 2.1.5. Analyseenhetene ble kodet for alle handlinger som kunne bli funnet, der det ofte ble observert flere handlinger i en enhet. For at en handling skulle bli identifisert må teksten av enheten direkte implisere at handlingen skal brukes av eleven. Videre blir det presentert en tabell over handlingene og beskrivelsen av disse fra Kapittel 2.1.5, samt et eksempel fra datamaterialet for hver handling.

Tabell 4.1. Koder for handlinger med beskrivelse og eksempel fra datamaterialet. Eksempler hentet fra Matemagisk (Kongsnes & Wallace, 2022) og Maximum (Tofteberg et al., 2021).

Handling	Beskrivelse	Eksempel fra datamaterialet
a) Følge en prosedyre	Følge instruksjoner stegvis, repetere eller fortsette et mønster	 Hentet fra Kongsnes & Wallace, 2022, s. 96

b) Finne regel	Finn ut prosedyren, regelen eller mønsteret som genererer et utfall, for eksempel en tallsekvens	 Hentet fra Kongsnes & Wallace, 2022, s. 104
c) Feilsøke	Feilsøk en kode	Finnes ikke eksempel i datamaterialet
d) Forme og skape	Gi instruksjoner, lage mønstre, skrive kode, representer med symboler	**2.72** **Loddsalg og trekning** En blokk med lynlodd inneholder 1000 lodd, nummerert fra 000 til 999. En speidergruppe har loddsalg og selger i alt fem slike blokker. Bøkene er merket med bokstavene A–E. Det finnes 3 hovedgevinster, 50 andre-premier og 150 trøstepremier. Lag et dataprogram som trekker gevinstene tilfeldig. Programmet skal oppgi loddnumrene som vinner. Hentet fra Tofteberg et al., 2021, s. 140
e) Forklare	Forklare med naturlig språk, bruke ord for å beskrive en prosedyre, en regel, et mønster eller et konsept	 Hentet fra Kongsnes & Wallace, 2022, s. 94-95 Se vedlegg 1 for større figur
f) Forestille seg	Forutse hva som vil skje, reflektere over mulige utfall når betingelser og verdier blir forandret	**2.48** Samarbeid to og to eller tre og tre. Ta utgangspunkt i situasjonen som oppgaven ovenfor beskriver. Sett først opp en hypotese: Hvor mange rette er det mest sannsynlig at eleven som tipper, vil få? Begrunn hypotesen deres. Lag et dataprogram som simulerer situasjonen. Kjør 500 simuleringer, og finn ut hvor mange rette det er mest sannsynlig at eleven som tipper, vil få. Hvor godt stemte hypotesen deres? Hentet fra Tofteberg et al., 2021, s. 117

Videre skal jeg presentere hvordan jeg har tolket og identifisert handlingene, samt hvordan jeg har adskilt dem fra hverandre. Under analyseprosessen av handlinger kom det frem at man enten kunne kode handling *b) Finne regel* eller *e) Forklare* ut fra hvordan man tolket analyseenheten. Her valgte jeg å se på hvor mye som forventes av elevene, der man i *b)* er mer målrettet på å lete etter en prosedyre eller regel mens man i *e)* går mer i dybden og elevene må lage en forklaring som krever mer av dem enn i *b)* handlingen. Denne distinksjonen vil man se et eksempel på i analysen av den spesielle analyseenheten fra Matemagisk, som blir presentert i Kapittel 4.1.4.

Jeg opplevde også at handlingene *a) Følge en prosedyre* og *d) Forme og skape* kunne bli kodet til samme analyseenhet avhengig av hvordan jeg tolket handlingene. Begge handlingene innebærer å skrive kode, men igjen, som med distinksjonen av *b)* og *d)*, har jeg kodet dem ut fra hva og hvor mye som forventes av eleven. Om eleven skal fortsette eller gjøre om på en eksisterende kode har jeg tolket det som handling *a)*, mens om eleven må bygge opp og skrive hele koden selv har jeg kodet det som *d)*. En analyseenhet blir i tillegg tolket som *a)* istedenfor *d)* hvis elevene blir bedt om å lage et program, men har tilgang til en allerede laget kode fra et eksempel eller tidligere oppgave som de kan endre eller utvide.

Til slutt har jeg kodet handling *f) Forestille seg* når elevene blir bedt om å tenke over hva de forventet før de utførte oppgaven eller blir bedt om å reflektere over resultatene de fikk. I tillegg ble handling *c) Feilsøke* kodet hvis elevene ble bedt om å finne noe feil ved et program. Om dette var en programmeringsfeil eller en feil som gjorde at matematikken brukt i koden ikke ga riktig resultat, skulle det merkes som denne handlingen.

4.1.2 Delanalyse 2: Begreper

Under analysen av begreper ble det, som Bråting og Kilhamn (2021) gjorde i sin studie, identifisert og klassifisert hvilke matematikk- og programmeringsbegrep som ble brukt i analyseenhetene. Ved å se på hvilke begrep som brukes i lærebøkene får man innsikt i hva som settes i fokus og vektlegges i lærebøkene innenfor sannsynlighetstemaet og algoritmisk tenkning. Begrepene ble identifisert og klassifisert innenfor *matematiske begrep* eller *programmeringsbegrep* ut fra ordbruk og meningen bak dem i kontekst av analyseenheten. Hvert begrep ble bare telt 1 gang per analyseenhet, selv om det ble brukt flere ganger. Begrep ble bare klassifisert hvis de ble brukt eksplisitt i analyseenheten eller kan bli funnet i tilhørende figurer eller liknende.

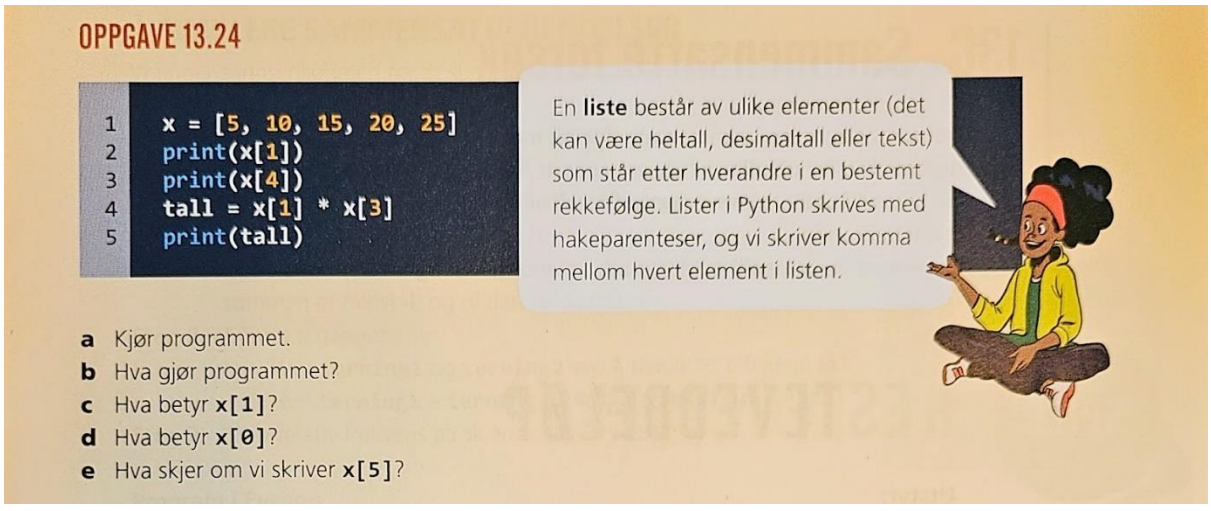
Innenfor programmeringsbegrep, ble begrepene Bråting og Kilhamn (2021) fant i sin studie, samt andre som ble funnet under analysen, kodet. Begrepene *simulere/simulering* og *program* dukket opp i en stor del av enhetene og ble dermed valgt å klassifiseres på lik linje som de originale begrepene Bråting og Kilhamn (2021) fant. *Simulere/simulering* ble kodet som et programmeringsbegrep, da analyseenhetene brukte det på den måten at elevene skulle for eksempel simulere terningkast ved hjelp av programmering. Et annet eksempel på dette kan vi finne i Figur 4.1, der elevene blir bedt om å lage et dataprogram som simulerer en viss situasjon. Denne situasjonen skal også kjøres gjennom 500 simuleringer, som i programmering krever at en løkke brukes, som gjør at vi ser på *simulering* i en programmeringskontekst. Begrepet *program* eller liknende, som i Figur 4.1 der vi finner *dataprogram*, brukes i den forstand at et program er en kode som elevene enten skal kopiere, reflektere over eller skrive selv.

2.48 Samarbeid to og to eller tre og tre. Ta utgangspunkt i situasjonen som oppgaven ovenfor beskriver. Sett først opp en hypotese: Hvor mange rette er det mest sannsynlig at eleven som tipper, vil få? Begrunn hypotesen deres.

Lag et dataprogram som simulerer situasjonen. Kjør 500 simuleringer, og finn ut hvor mange rette det er mest sannsynlig at eleven som tipper, vil få. Hvor godt stemte hypotesen deres?

Figur 4.1. Eksempel på analyseenhet som inneholder programmeringsbegrepene *simulere/simulering* og *program* fra Maximum (Tofteberg et al., 2021, s. 117).

Programmeringsbegrepene Bråting og Kilhamn (2021) utarbeidet er, som nevnt i Kapittel 2.2.4, *algoritme/stegvise instruksjoner*, *løkke/iterasjon/repetisjon*, *regel*, *kode*, *betingelse* og *feil/feilsøk*. I tillegg til eksplisitte funn av begrepene i teksten til analyseenheten, skal jeg nå opplyse om kriteriene for implisitte funn av begrepene som førte til at disse ble identifisert. *Algoritme/stegvise instruksjoner* ble kodet hvis elevene ble gitt en kode som inneholdt en algoritme eller ble gitt noen stegvise instruksjoner for hvordan de burde gå frem, som i Figur 4.2. Begrepet *løkke/iterasjon/repetisjon* og *betingelse* ble kodet hvis man kunne finne en løkke eller if-setning respektivt i gitte koder. *Regel* ble kodet hvis elevene i analyseenheten ble forklart et moment eller en type regel, som i Figur 4.2 der en tilhørende snakkeboble til enheten forklarer hva en liste er i Python. *Kode* ble kodet hvis dette begrepet ble brukt og ikke om det ble gitt en kode eller program i analyseenheten grunnet et overlapp med begrepene *algoritme/stegvise instruksjoner* og *program*. Til slutt ble *feil/feilsøke* kodet hvis elevene ble bedt om å finne en feil i gitt eller skrevet program i løpet av analyseenheten.



Figur 4.2. Eksempel på analyseenhet som inneholder programmeringsbegrepene *program*, *algoritme/stegvise instruksjoner* og *regel* fra Matemagisk (Kongsnes & Wallace, 2022, s. 101).

Som nevnt i Kapittel 3.1, måtte jeg jobbe induktivt i utformingen av kategorier for de matematiske begrepene, da jeg ikke kunne ta utgangspunkt i begrepene til Bråting og Kilhamn (2021). De har hovedsakelig sett på algebraoppgaver i lærebøkene for barnetrinnet, mens jeg har sett på sannsynlighetstemaet i lærebøkene for 9. trinn. Jeg startet med å identifisere alle begreper som hadde en matematisk mening i kontekst av analyseenheten, som for eksempel *sannsynlighet*, *terning*, *mindre enn*, *tipping*, *prosent*, og *forventet*, som kan ses i Tabell 4.2. Etter at alle analyseenhetene hadde blitt gått gjennom, slo jeg sammen begreper som har samme betydning, der *sannsynlig*, *sannsynlighet*, *eksperimentell sannsynlighet* ble til *sannsynlighet*, og *terningkast*, *terning*, *kast*, *kuler*, *mynt*, *myntkast*, *lodd*, *premie* og *tipping* som sammen ble til *random device*. Videre ble begrepene fordelt i underkategorier som jeg anså som passende, som vist i Tabell 4.3. Til slutt ble alle analyseenhetene gått over igjen, der det ble telt opp hvilke underkategorier av begreper som ble identifisert. Hver analyseenhet kan inneholde flere underkategorier, men hver kategori ble bare telt opp en gang per enhet. For eksempel inneholder analyseenheten i Figur 4.1 de matematiske begrepene *sannsynlighet* i teksten og *tipping* fra oppgaven ovenfor. *Tipping* går under *random device*, og både *sannsynlighet* og *random device* tilhører underkategorien *sannsynlighetsbegreper*. Dermed blir denne kategorien klassifisert og telt opp en gang. Det er verdt å merke seg at det ikke blir gjort noen funn av matematiske begreper i Figur 4.2, der dette er en av få enheter med programmering fullt i fokus.

Tabell 4.2. Matematiske begrep før sammenslåing og underkategorisering. Alle begrepene er satt i rekkefølge etter antall ganger de blir brukt og fordelt ut fra hvilken lærebok de ble identifisert i.

Matemagisk	**Maximum**
Sannsynlighet 10	*Sannsynlighet 5*
Relativ frekvens 9	*Terningkast 3*
Terning 5	*Trekning 3*
Sum 4	*Utfall 3*
Teller 3	*Antall 2*
Terningkast 3	*Tilfeldig 2*
Utfall 3	*Variasjon 2*
Kast 2	*Data 1*
Like 2	*Eksperimentell sannsynlighet 1*
Mindre enn 2	*Forventet 1*
Forventet andel 1	*Frekvens 1*
Hendelser 1	*Gjennomsnitt 1*
Kuler 1	*Gjennomsnittsverdi 1*

Mer 1	*Hendelser 1*
Minst 1	*Kombinasjoner 1*
Mynt 1	*Lodd 1*
Oddetall 1	*Myntkast 1*
Partall 1	*Opptelling 1*
Prosent 1	*Premie 1*
Sammensatte hendelser 1	*Rekke 1*
Sammensatte forsøk 1	*Sammenlikn 1*
Tilbakelegging 1	*Sannsynlig 1*
Tilnærmet lik 1	*Sum 1*
Trekning 1	*Summering 1*
Valg 1	*Tell opp 1*
Vinnersjanse 1	*Tilbakelegging 1*
	Tipping 1

Tabell 4.3. Matematiske begrep etter sammenslåing og underkategorisering. Begrepene er satt i rekkefølge etter antall ganger de blir brukt og fordelt ut fra hvilken lærebok de ble identifisert i.

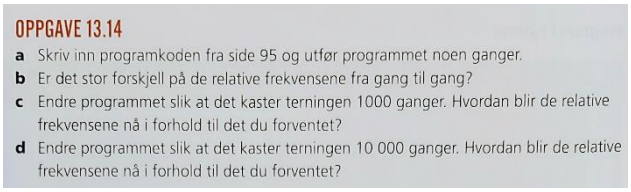
Matemagisk	**Maximum**	**Underkategorier**
Random device 12	*Sannsynlighet 8*	Sannsynlighetsbegreper
Sannsynlighet 10	*Random device 7*	
Relativ frekvens 9	*Trekning 3*	
Utfall 3	*Utfall 3*	
Forventet andel 1	*Gjennomsnitt 2*	
Hendelser 1	*Tilfeldig 2*	
Sammensatte hendelser 1	*Variasjon 2*	
Sammensatte forsøk 1	*Forventet 1*	
Tilbakelegging 1	*Frekvens 1*	
Trekning 1	*Hendelser 1*	
Valg 1	*Kombinasjoner 1*	

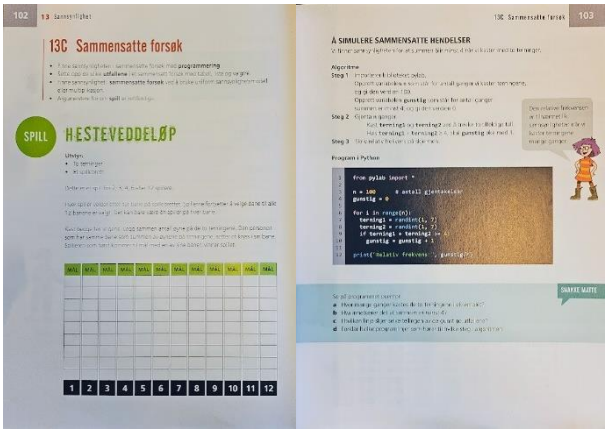
Vinnersjanse 1	*Rekke 1* *Tilbakelegging 1*	
Like 2 *Mindre enn 2* *Mer 1* *Minst 1* *Oddetall 1* *Partall 1* *Tilnærmet lik 1*	*Antall 2*	Mengdebegreper
Sum 4 *Opptelling 3* *Prosent 1*	*Sum 2* *Opptelling 2*	Generelle matematiske begreper
	Data 1	Statistikk begreper

4.1.3 Delanalyse 3: Læring av sannsynlighet

Under delanalysen om læring av sannsynlighet gikk jeg ut fra min syntese av Castro (1998) og Wilensky (1995) sine prinsipper, der forklaringsspørsmålene ble brukt som hjelp for å kode de forskjellige kategoriene. I Tabell 4.4 blir kategoriene fra min modell presentert med forklaringsspørsmål fra Kapittel 2.3.4, med et eksempel fra datamaterialet for hver kategori.

Tabell 4.4. Koder for kategorier med beskrivelse og eksempel fra datamaterialet. Eksempler hentet fra Matemagisk (Kongsnes & Wallace, 2022) og Maximum (Tofteberg et al., 2021).

Kategori	**Forklaringsspørsmål**	**Eksempel fra datamaterialet**
1. Elevenes tankegang og intuisjon	Blir elevenes intuisjon og forforståelse utfordret i oppgaven? Blir det tatt opp misoppfatninger innenfor sannsynlighet?	 Hentet fra Kongsnes & Wallace, 2022, s. 96

2. Knytte virkelighet og teori sammen	Har oppgavene et empirisk fokus eller er virkelighetsnære? Blir det stilt spørsmål til elevene som kan utvikle deres sannsynlighetskunnskap ved hjelp av empiri/virkeligheten?	**2.52** Samarbeid to og to eller tre og tre. Lag et dataprogram som kan foreta en trekning. Den som bruker programmet, skal oppgi totalt antall tall det skal trekkes fra, hvor mange tall som skal trekkes, og om trekningen er med eller uten tilbakelegging. Spillet Lotto fra Norsk Tipping går ut på å tippe 7 vinnertall av 34 mulige tall. Trekningen skjer uten tilbakelegging. Bruk dataprogrammet dere har laget, til å trekke ut en lottorekke. Hentet fra Tofteberg et al., 2021, s. 120
3. Matematiske sammenhenger	Bygges det forbindelser mellom forskjellige sannsynlighetsbegreper i oppgavene? Blir forskjellige begreper satt opp mot hverandre og stilt spørsmål ved for å utvikle elevenes kunnskap videre?	 13C Sammensatte forsøk SPILL HESTEVEDDELØP Å SIMULERE SAMMENSATTE HENDELSER Hentet fra Kongsnes & Wallace, 2022, s. 103 Se vedlegg 2 for større figur
4. Oppsett	Hvordan skal oppgaven utføres? Hva slags verktøy skal benyttes?	**2.34** **a** Lag et dataprogram som simulerer et valgt antall terningkast, og som samtidig teller frekvensen av hvert utfall. **b** Endre dataprogrammet til å simulere et valgt antall terningkast med to terninger som summeres. Programmet skal fremdeles telle frekvensen av hvert utfall. Hentet fra Tofteberg et al., 2021, s. 109

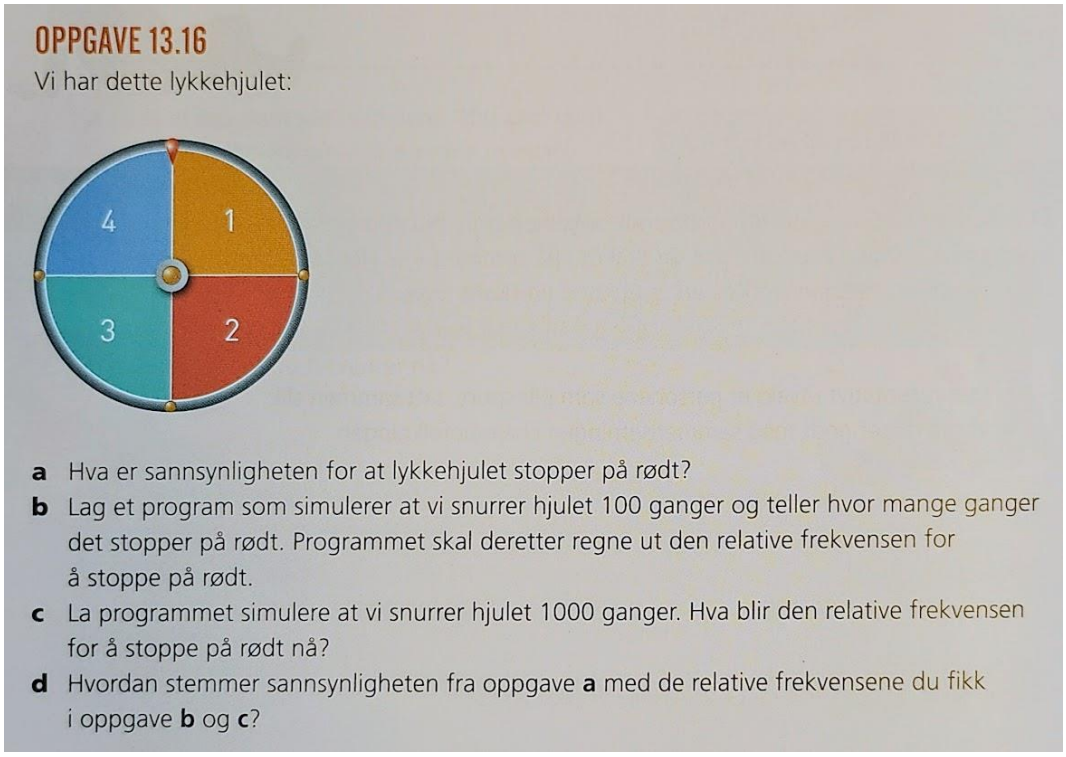
Kategori *1. Elevenes tankegang og intuisjon* ble kodet om analyseenheten ble bedt om å tenke over resultatene de fikk eller hva de forventet før de utførte analyseenheten. Kategori *2. Knytte virkelighet og teori sammen* ble kodet om analyseenheten tok utgangspunkt i hverdagskontekst som terningkast, spill, fødsel, etc. Kategori *3. Matematiske sammenhenger* ble kodet om analyseenheten inneholdt mer enn ett tema innenfor sannsynlighet og disse ble satt opp mot hverandre for å utvikle elevenes forståelse. Denne kategorien ble ikke kodet om det innebar sammenheng innenfor bare ett sannsynlighetstema. Kategori *4. Oppsett* ble kodet om analyseenheten spesifikt ønsket en viss type utførelse eller verktøy som skulle brukes. Da jeg spesifikt har plukket ut analyseenheter som inneholder programmering, vil majoriteten av enhetene inneholde denne kategorien siden verktøyet, som er programmering, dermed er fastsatt.

4.1.4 Eksempler fra lærebøkene

Videre vil jeg presentere 4 analyseenheter fra lærebøkene Matemagisk og Maximum. Under analyseprosessen observerte jeg noen oppgavetyper som gikk igjen i hver av lærebøkene, og anser da disse analyseenhetene som typiske for læreboka den er fra. Lærebøkene inneholdt også analyseenheter som skilte seg ut fra de typiske enhetene. Disse har jeg kalt for spesielle enheter da dette var enheter med en oppbygning eller innhold som elevene bare kom over noen få ganger kontra de typiske enhetene som til slutt ble gjenkjennbare for elevene. Det ble dermed valgt ut en typisk og en spesiell enhet fra hver lærebok for å presentere analyseprosessen og utvalget fra lærebøkene. I begge eksemplene på spesielle enheter får elevene introdusert en ferdig kode, som gjør det å sammenlikne disse analyseenhetene og hvordan lærebøkene går frem mer interessant da de har samme utgangspunkt, men spør om forskjellige ting. Valget mitt av å presentere en typisk og en spesiell enhet fra hver lærebok går ut fra at man da får sett hva som kjennetegner de to lærebøkene i grove trekk, med tanke på struktur og innhold av enheter og hva de har ansett som viktig å trekke frem med de nye retningslinjene LK20 ga innenfor programmering og algoritmisk tenkning. I tillegg gir disse eksemplene en illustrasjon av analyseprosessen og hvordan jeg har jobbet med datamaterialet. Jeg vil først presentere de typiske enhetene fra Matemagisk og Maximum og analysen av disse før de spesielle enhetene blir presentert og analysert.

For hver analyseenhet vil delanalysene med klassifisering og prosess bli gjengitt. Delanalysene vil bli presentert i hvert sitt avsnitt i rekkefølgen de ble presentert i tidligere i Kapittel 4.1. Først vil analysen av handlingene presenteres, så begreper og til slutt kategorier.

Typisk analyseenhet fra Matemagisk



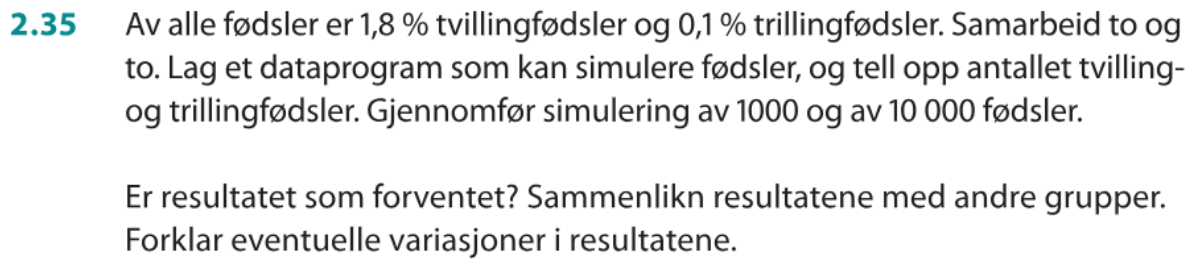
Figur 4.3. Typisk analyseenhet fra Matemagisk (Kongsnes & Wallace, 2022, s. 98).

I Matemagisk observerte jeg at 13 av 16 (81%) enheter inneholdt deloppgaver, der elevene først enten blir bedt om å lage eller kopiere et program for så å endre på dette videre i oppgaven. De blir ofte presentert for en type *random device*, et virkelighetsnært eksperiment som skal gi resultatene en kontekst. Figur 4.3 viser en typisk enhet med denne oppbygningen fra Matemagisk. Enhetene vil noen ganger inneholde en deloppgave som ikke omhandler programmering, som deloppgave a) i Figur 4.3, men anses ikke som typisk i datamaterialet fra Matemagisk. Her blir elevene bedt om å lage et program i deloppgave b), som vi gjenkjenner som handling *d) Forme og skape*. Videre skal elevene i deloppgave c) endre på en parameter og kjøre koden på nytt, noe jeg har kodet som handling *a) Følge en prosedyre*. Til slutt blir de i deloppgave d) bedt om å reflektere over sannsynlighetene de har fått i de tidligere deloppgavene, noe som gir oss handling *f) Forestille seg*.

Programmeringsbegrepene som man kan identifisere eksplisitt i enheten er *program* og *simulere*. Her er *simulere* valgt som et programmeringsbegrep da det er i konteksten at det er programmet som utfører simulasjonen. Av matematiske begreper finner vi *sannsynlighet*, *teller* og *relativ frekvens*, som alle klassifiseres i underkategorien *sannsynlighetsbegreper*.

Til slutt i delanalysen om læring av sannsynlighet finner man kategori *4. oppsett* ved at det skal utføres ved hjelp av programmering, kategori *2. knytte virkelighet og teori sammen* med at oppgaven tar utgangspunkt i et lykkehjul og sannsynligheten for hvor lykkehjulet stopper, samt kategori *1. elevenes tankegang og intuisjon* med at de i deloppgave d) blir bedt om å reflektere over det de har regnet ut og det programmet kom frem til.

Typisk analyseenhet fra Maximum


2.35 Av alle fødsler er 1,8 % tvillingfødsler og 0,1 % trillingfødsler. Samarbeid to og to. Lag et dataprogram som kan simulere fødsler, og tell opp antallet tvilling- og trillingfødsler. Gjennomfør simulering av 1000 og av 10 000 fødsler.

Er resultatet som forventet? Sammenlikn resultatene med andre grupper. Forklar eventuelle variasjoner i resultatene.

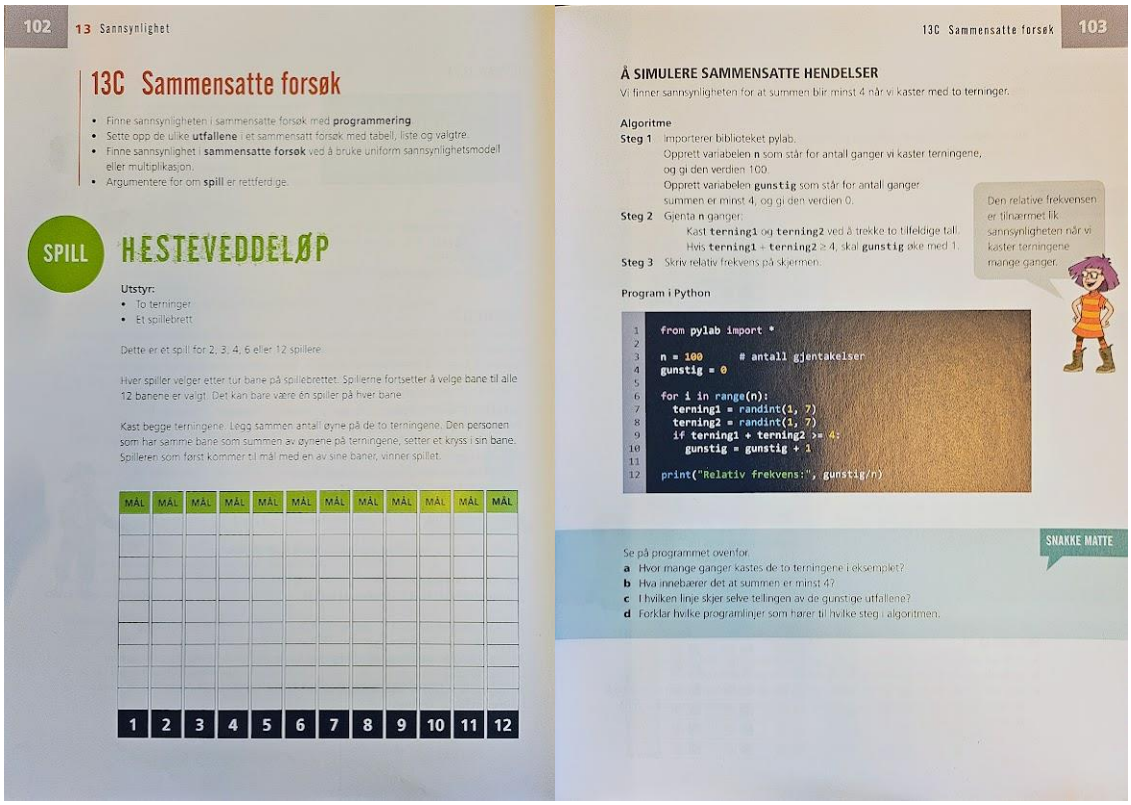
Figur 4.4. Typisk analyseenhet fra Maximum (Tofteberg et al., 2021, s. 109).

I Maximum preges enhetene av at elevene får et større og mer åpent spørsmål, som ikke har blitt dekomponert inn i deloppgaver. Elevene blir ofte bedt om å diskutere og sammenlikne resultatene sine med andre i klassen. Dette kan vi se et eksempel på i Figur 4.4, der elevene blir bedt om å lage et eget program, som vi igjen gjenkjenner som handling *d) forme og skape*. I tillegg blir elevene bedt om å ta stilling til om resultatet de fikk var det de forventet og sammenlikne med andre grupper, samt å forklare eventuelle forskjeller i resultatene, noe jeg har kodet til handling *f) forestille seg*.

På samme måte som i den typiske analyseenheten fra Matemagisk finner man *dataprogram*, som blir klassifisert som *program*, og *simulere* som programmeringsbegreper benyttet i enheten. Av matematiske begreper ble *tell opp*, *forventet* og *variasjon* identifisert, der *forventet* og *variasjon* kan dras sammen til *sannsynlighetsbegreper*, mens *tell opp* går under *generelle matematiske begreper.*

Igjen finner vi de samme kategoriene som i den typiske oppgaven fra Matemagisk. Det er spesifikt programmering som er verktøyet, som gir oss kategori *4. oppsett*, mens kategori *2. knytte virkelighet og teori sammen* kommer fra oppgavens grunnlag i tvilling- og trillingfødsler. Til slutt ser vi kategori *1. elevenes tankegang og intuisjon* i at elevene blir bedt om å reflektere over resultatene sine og diskutere med andre om hva de har fått og hva de forventet.

Spesiell analyseenhet fra Matemagisk



Figur 4.5. Spesiell analyseenhet fra Matemagisk (Kongsnes & Wallace, 2022, s. 103). For større figur se Vedlegg 2.

I denne mer spesielle analyseenheten fra Matemagisk blir elevene introdusert til spillet «hesteveddeløp», samt en algoritme og program i Python for å simulere sammensatte hendelser. Enheten skiller seg ut fra de mer typiske enhetene for Matemagisk i omfanget av informasjon elevene blir gitt og det faktum at de ikke blir bedt om å kopiere eller skrive kode selv, men forklare ulike momenter og vise forståelse for hvordan et program er bygd opp og kjører. Istedenfor å utføre noe er det elevenes forståelse som er sentralt i denne enheten.

Elevene blir i deloppgave a) og c) bedt om å finne i programmet hvor en spesifikk variabel er og hvor selve tellingen skjer, som jeg har kodet til handling *b) finne regel*. I deloppgave b) og d) derimot skal elevene gå mer i dybden og forklare hva som skjer i koden, samt forklare hvorfor summen må være minst 4, som jeg har kodet til handling *e) forklare*. En forskjell mellom disse deloppgavene er hvor mye arbeid som kreves av eleven, noe jeg bruker for å skille mellom handlingene *b) finne regel* og *e) forklare*.

Av programmeringsbegreper finner vi *algoritme* og *løkke* i gitt kode, *betingelse* i if-setning i gitt kode og *program*. Innenfor matematikk finner vi begreper som *sannsynlighet*, *sum*, *terning*, *sammensatte hendelser*, *relativ frekvens*, *tilnærmet lik* og *gunstig utfall*. Her vil *sannsynlighet*, *terning*, *sammensatte hendelser*, *relativ frekvens* og *gunstig utfall* samles under *sannsynlighetsbegreper*, mens *tilnærmet lik* er et begrep om *mengde*, og til slutt vil *sum* gå under *generelle matematiske begreper*.

Hvordan oppgaven skal utføres er fastsatt som gir oss kategori *4. oppsett*. Videre får vi kategori *2. knytte virkelighet og teori sammen* fra utgangspunktet i spillet «hesteveddeløp». Til slutt ser vi at prøver å vise en sammenheng mellom relativ frekvens og sannsynligheten når antall gjentakelser blir høy og hvordan de da blir tilnærmet like, som gir oss kategori *3. matematiske sammenhenger*.

Spesiell analyseenhet fra Maximum

OPPDRAG

Forklare store talls lov med mange eksperimenter

Du har kanskje verken tid eller lyst til å kaste en mynt eller terning tusenvis av ganger? Datamaskinen er et godt verktøy når vi skal gjenta samme handling flere ganger. Ved hjelp av programmering skal vi se hvordan store talls lov fungerer. Utgangspunktet er situasjonen med myntkast som vi studerte på forrige side. Nå skal vi få en datamaskin til å gjenta myntkastet veldig mange ganger og finne gjennomsnittet.

Del 1 – Myntkastet

1 Vi må først definere at krone = 1, mynt = 0. Hva forventer du at gjennomsnittsverdien blir hvis du kaster mynt og krone mange ganger?

2 Bruk programmet i Python til å gjennomføre 10 kast. Se hvordan gjennomsnittet utvikler seg.

3 Endre verdien av *N* til 50, 100 og 1000. Beskriv hvordan kurven utvikler seg.

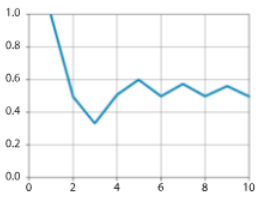
Del 2 – Summen av to terningkast

1 Lag en oversiktstabell som viser alle mulige utfall av summen av to terningkast. Bruk tabellen til å sette opp en hypotese over hvilken sum dere tror vil være et gjennomsnittlig resultat for mange kast med to terninger.

2 Gjør endringer i programmet slik at det kan gjennomføre to terningkast med verdier 1–6 og summere disse. Lag plot for 100, 1000 og 10 000 doble terningkast.

3 Stemmer hypotesen deres?

```
from pylab import *

# N = antall myntkast
N=10

# Lager en liste med N myntkast,
# hvert kast får verdien 0 eller 1
kast=randint(0,2,N)

# lager en liste med kastnummer (x-verdier)
x=arange (1,N+1)

# lager liste med kumulativt
# gjennomsnitt (y-verdier)
gjsnitt_kast=cumsum(kast)/x

plot(x,gjsnitt_kast)
axis([0,N,0,1])
grid()
show()
```



Eksempel på resultat etter 10 myntkast.

Figur 4.6. Spesiell analyseenhet fra Maximum (Tofteberg et al., 2021, s. 101). For større figur se Vedlegg 3.

I dette oppdraget skal elevene utforske hvordan store talls lov fungerer ved hjelp av programmering. På samme måte som den spesielle enheten fra Matemagisk, skiller denne spesielle enheten fra Maximum seg fra resten av datamaterialet med mengden informasjon elevene får i løpet av oppdraget. Dette er i tillegg 1 av 2 enheter som gir elevene et ferdig program som de skal gå ut fra i de 14 analyseenhetene fra Maximum. Datamaterialet fra Maximum kan stille spørsmål til elevenes forventninger og resultater, og blir ikke ansett som noe spesielt i denne læreboka. Mengden deloppgaver skiller seg ut der analyseenhetene fra Maximum typisk har en utforming på enhetene uten å dekomponere spørsmålene i deloppgaver.

Oppdraget er delt i to deler: del 1 der det tas for seg myntkast og del 2 om sum av to terningkast. I første deloppgave på både del 1 og 2, samt deloppgave 3 i del 2, blir elevene bedt om å reflektere over hva de forventer, som gir oss handling *f) Forestille seg*. Videre finner vi handling *a) Følge en prosedyre* i deloppgave 2 og 3 i del 1 og deloppgave 2 i del 2, der elevene blir bedt om å kjøre gitt kode med ulike parametere. Til sist finner vi handling *e) forklare* i del 1 deloppgave 3 der de skal beskrive utviklingen av kurven.

Av programmeringsbegreper finner vi *programmering* som blir kodet som *program* og man kan finne *algoritme* i gitt kode som viser hvordan man kan bygge opp en kode som kan gjøre det som er ønsket. I de matematiske begrepene finner vi *myntkast*, *gjennomsnitt*, *gjennomsnittsverdi*, *mulig utfall* og *sum*. *Myntkast* vil gå under *random device* begrepet, og vil sammen med *mulig utfall, gjennomsnitt* og *gjennomsnittsverdi* bli satt i *sannsynlighetsbegreper*. *Gjennomsnitt* og *gjennomsnittsverdi* vil falle inn under *sannsynlighetsbegreper* grunnet meningen enheten gir begrepene i kontekst av forventningsverdi og store talls lov. Videre vil *sum* falle inn under *generelle matematiske begreper*.

Oppsettet og verktøyene som skal brukes er klart i oppgaven, som gir oss kategori *4. oppsett* i den tredje delen av analysen. I tillegg får vi kategori *2. knytte virkelighet og teori sammen* fra oppgavens utgangspunkt i mynt- og terningkast, samt kategori *1. elevenes tankegang og intuisjon* i at elevene skal tenke over hva de forventer og hva de har fått av resultater i løpet av oppgaven.

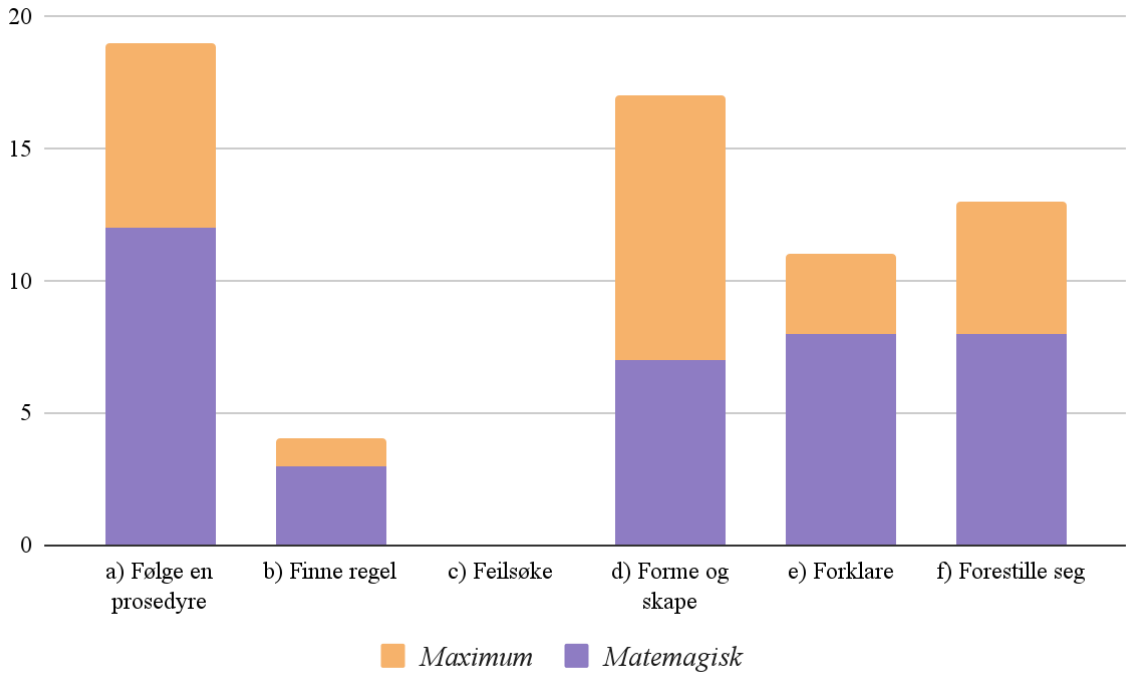
4.2 Funn fra analysen

Videre vil funn fra de forskjellige delanalysene handlinger, begreper og læring av sannsynlighet presenteres individuelt.

4.2.1 Handlinger

Funn 1: Ingen funn av *c) Feilsøke*

Figur 4.7 viser fordelingen over funn av handlinger i de to lærebøkene. Gjennom analyseprosessen ble ikke handling *c) Feilsøke* funnet i noen av analyseenhetene fra de to lærebøkene. Viktigheten og betydningen av Funn 1 vil diskuteres videre i Kapittel 5.1.



Figur 4.7. Diagram som viser fordelingen over handlinger med utgangspunkt i rammeverket til Bråting og Kilhamn (2021) funnet i Maximum og Matemagisk.

Funn 2: Ulik fordeling av handling *a)* og *d)* i de to lærebøkene

De to handlingene som det ble funnet flest av totalt er også de som det varierer mest i mengde av i de to lærebøkene. Handling *a) Følge en prosedyre* ble kodet til 12 av 16 (75%) analyseenheter i Matemagisk og 7 av 14 (50%) analyseenheter i Maximum. I kontrast ble handling *d) Forme og skape* kodet til 7 av 16 (44%) oppgaver i Matemagisk og 10 av 14 (71%) oppgaver i Maximum. Man ser dermed en ulik fordeling av handlinger i de to lærebøkene, der Matemagisk har et fokus på handling *a)* mens Maximum har sin største bolk av handlinger i handling *d)*. Et eksempel på dette kan man finne i de typiske analyseenhetene fra lærebøkene i Kapittel 4.1.4 og handlingene funnet i disse.

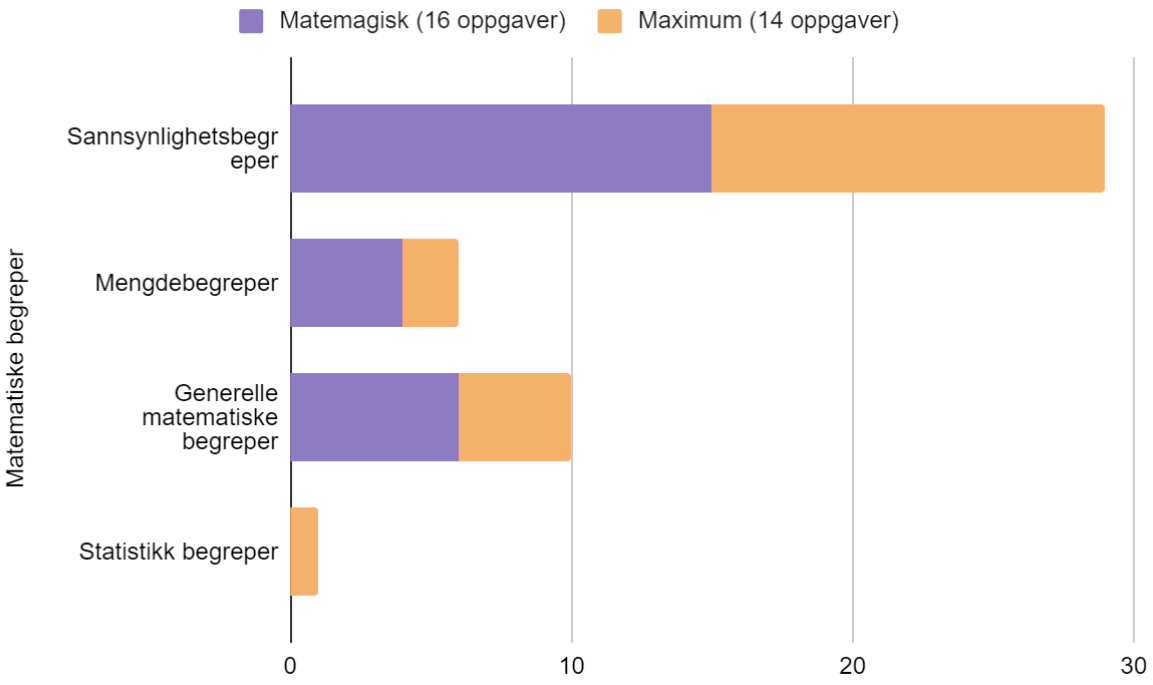
Samtidig inneholder 13 av 16 (83%) av analyseenhetene fra Matemagisk flere deloppgaver i motsetning til 8 av 14 (57%) av enhetene i Maximum, som heller stiller et større og mer åpent spørsmål til elevene. Jeg vil også bemerke at enhetene med deloppgaver i Maximum inneholder 2-3 av disse, kontra Matemagisk som typisk har 3-4 i hver enhet. Da kan det være naturlig å finne flere av handling *a) Følge en prosedyre* i Matemagisk da det blir gitt flere små deloppgaver kontra en åpen oppgave der elevene blir bedt om å lage et program som man ofte ser i analyseenhetene fra Maximum. Det virker dermed som at Matemagisk bryter ned og dekomponerer et større arbeid til mindre deloppgaver, mens Maximum stiller mer åpne og vide spørsmål i de analyserte enhetene innenfor sannsynlighetskapittelet i lærebøkene.

4.2.2 Begreper

Funn 3: Størst andel sannsynlighetsbegreper innenfor matematiske begreper

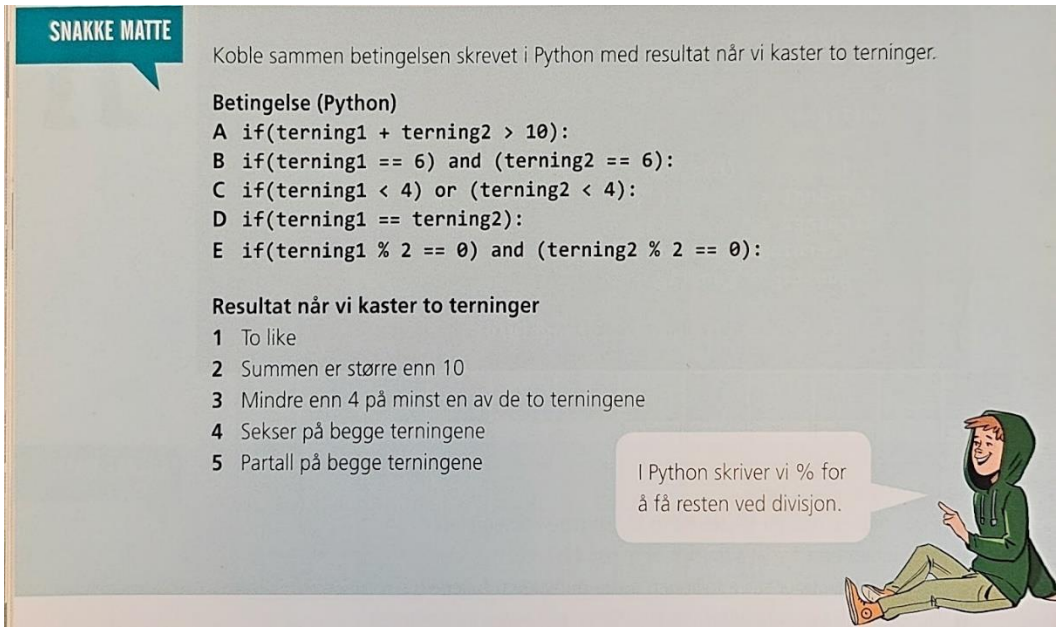
Figur 4.8 viser fordelingen over matematiske begreper funnet i de to lærebøkene. Nesten alle analyseenhetene inneholder et type begrep innenfor sannsynlighet, der det bare er en enhet fra Matemagisk som ikke inneholder et sannsynlighetsbegrep. Denne analyseenheten er en introduksjon av lister, se Figur 4.2, og er den eneste enheten som bare omhandler programmering i datamaterialet. Det virker dermed som at begge lærebøkene har hatt som mål å integrere programmering og sannsynlighet sammen der de anser det som hensiktsmessig.

Samtidig, hvis vi ser på totalt antall enheter i hele sannsynlighetskapittelet i begge lærebøkene, både de med og uten programmering, inneholder 16 av 60 (27%) enheter i Matemagisk og 14 av 114 (12%) enheter i Maximum programmering. Enheter i bøkene som dermed svarer på læreplanmålet om at eleven skal "simulere utfall i tilfeldige forsøk og beregne sannsynligheten for at noe skal inntreffe, ved å bruke programmering", som er ett av to læreplanmål innenfor sannsynlighet for 9. trinn, er i minoritet. Det er i gjennomsnitt 20% av sannsynlighetskapitlene i lærebøkene som adresserer dette læreplanmålet om programmering. Det kan derfor virke som at man ønsker at den tradisjonelle matematikken skal stå i fokus, noe som kan ha ført til at programmering blir brukt som et verktøy for matematikk istedenfor et eget konsept.



Figur 4.8. Matematiske begreper identifisert og klassifisert ut fra den induktive analysen av datamaterialet etter rammeverket om begreper videreutviklet fra Bråting og Kilhamn (2021).

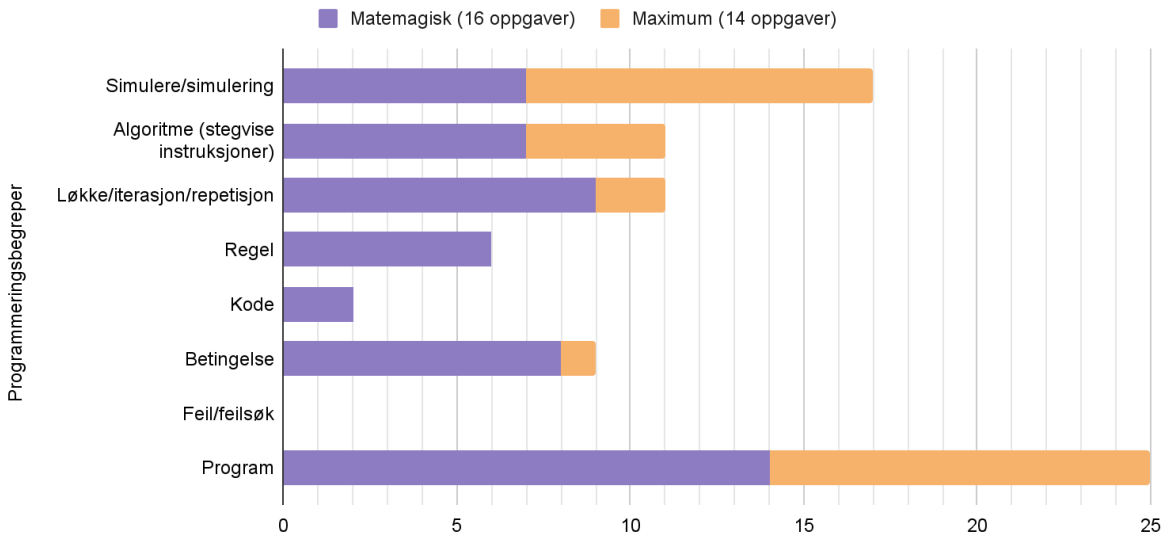
Innenfor de andre underkategoriene ble det gjort færre funn av begreper, som gir mening da dette kapittelet spesifikt handler om sannsynlighet. Samtidig er, for eksempel, generelle matematiske begreper som *tell opp* og *sum* funnet i de fleste enheter fra datamaterialet. Disse begrepene omhandler å skaffe data for å gi en større forståelse av sannsynligheten for ulike hendelser når de gjentas et visst antall ganger. Dette er ikke så overraskende siden mange sannsynlighets relevante begreper viser seg ved relative størrelser, som innebærer oppsummering først. For mengdebegrepene funnet, igjen om gitt antall og størrelse, ble majoriteten av disse observert i en analyseenhet fra Matemagisk, vist i Figur 4.9, under den induktive analysen av begreper i analyseenhetene. I Figur 4.9 ser vi at elevene får i oppgave å koble sammen betingelser skrevet i Python med ulike resultat når vi kaster to terninger. Her blir mengdebegreper som *like*, *større* og *mindre* brukt for å få en forståelse for hvordan man kan skrive disse forskjellige matematiske resultatene i en kode ved bruk av betingelser som if-setninger, logiske uttrykk som ==, %, < og > og hvordan dette blir tolket av Python.



Figur 4.9. Eksempel på analyseenhet som inneholder mengdebegreper og sannsynlighetsbegreper innenfor matematikk fra Matemagisk (Kongsnes & Wallace, 2022, s. 104).

Funn 4: Ulikt antall programmeringsbegreper funnet i lærebøkene

Under analysen av programmeringsbegreper identifisert i lærebøkene ble det raskt observert en forskjell i antall begreper mellom de to lærebøkene, se Figur 4.10. Totalt ble det funnet 53 programmeringsbegreper i Matemagisk kontra 28 i Maximum.



Figur 4.10. Programmeringsbegreper ut fra det tilpassede og utvidede rammeverket om begreper basert på Bråting og Kilhamn (2021).

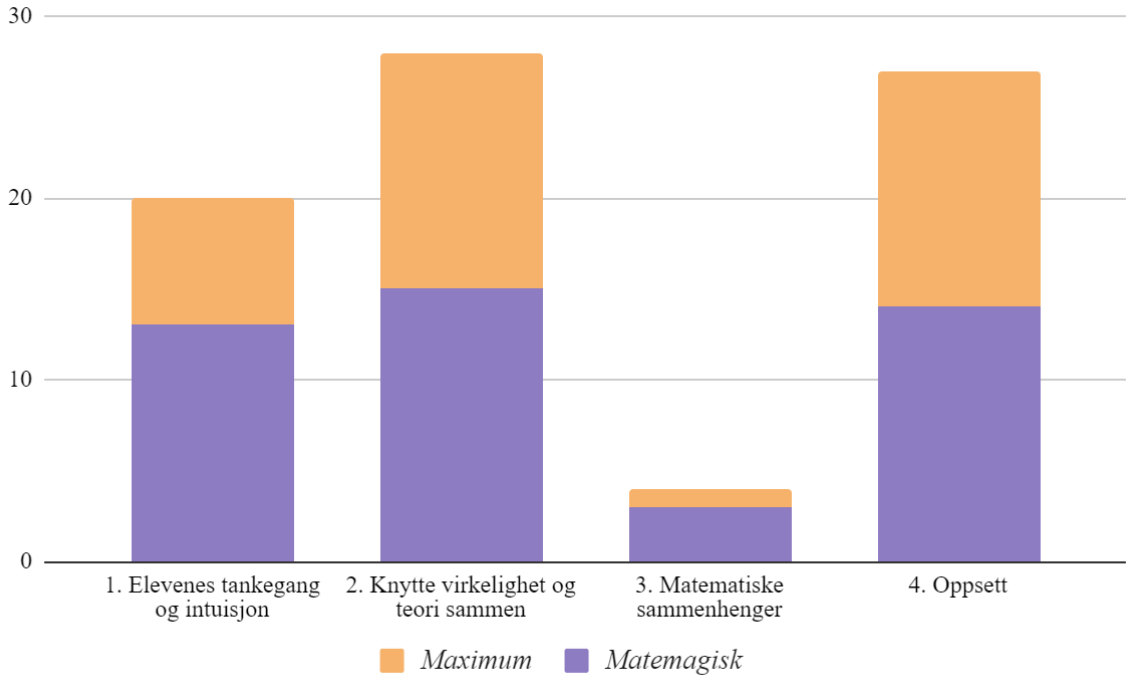
I begge lærebøkene ser vi at *simulere/simulering* og *program* er begreper som ofte blir identifisert. I tillegg inneholder Matemagisk flere enheter med begrepene *algoritme*, *løkke*, *regel* og *betingelse*. *Kode* er funnet eksplisitt som et begrep i Matemagisk, men brukes mer som et synonym for *program*, der begrepet *program* allerede har blitt brukt. Man kunne dermed sette sammen disse begrepene da de har samme betydning. Maximum inneholder færre begrep eksplisitt brukt, og inneholder færre tilhørende eksempler og koder til analyseenhetene enn Matemagisk. Generelt har jeg observert at analyseenhetene fra Maximum er ofte mer åpne og gir mindre retningslinjer for å løse dem, noe man også kan se ut fra handlingene identifisert i Kapittel 4.2.1, der Maximum inneholder flere enheter med *d) Forme og skape*, kontra Matemagisk med et større antall enheter med a*) Følge en prosedyre*. Dette kan forklare mangelen på programmeringsbegrepene jeg har klassifisert, og betyr ikke at Maximum inneholder oppgaver med mindre krav om programmerings- og CT-kunnskap eller er av mindre kvalitet. Derimot kan det være en ulik forventning til elevenes programmeringskunnskap, som skal diskuteres videre i Kapittel 5.2.

Jeg vil også bemerke mangelen i begge bøker på programmeringsbegrepet *feil/feilsøk*, som funnet i delanalysen av handlinger. Dette vil bli diskutert videre i Kapittel 5.1.

4.2.3 Læring av sannsynlighet

Funn 5: Lite funn av kategori *3. Matematiske sammenhenger*

Figur 4.11 viser fordelingen over kodede kategorier i de to lærebøkene. Man kan her se at det er gjort lite funn av kategori *3. Matematiske sammenhenger*. Totalt 4 av 30 (13%) analyseenheter ble kodet med denne kategorien, der 3 av disse var fra Matemagisk og 1 fra Maximum.



Figur 4.11. Diagram som viser fordelingen over kategorier ut fra egenutviklet modell om læring av sannsynlighet funnet i Matemagisk og Maximum.

Funn 6: Ulik fordeling av kategori *1.* i de to lærebøkene

Det ble observert en lik fordeling av kategoriene *2. Knytte virkelighet og teori sammen* og *4. Oppsett* i hver lærebok, som kan forklares av utvalget av programmeringsspesifikke enheter og fokuset på bruk av random device i sannsynlighetstemaet. Derimot var det en tydelig forskjell mellom lærebøkene når det gjelder den første kategorien. Kategori *1. Elevenes tankegang og intuisjon* ble kodet i 13 av 16 (81%) enheter i Matemagisk og 7 av 14 (50%) enheter i Maximum. Ut fra de valgte analyseenhetene ser man da at rundt 81% av enhetene fra Matemagisk, kontra 50% av enhetene fra Maximum, inneholder denne kategorien, som kan tyde på et større søkelys på elevenes intuisjon og tankegang i Matemagisk. Dette kan støttes opp mot flere funn av handlingene *e) Forklare* og *f) Forestille seg* i analyseenhetene fra Matemagisk kontra Maximum i delanalyse 1.

5 Drøfting

I dette kapittelet vil jeg diskutere funnene opp mot teori og læreplanen for å kunne svare på forskningsspørsmålene *(1) Hva karakteriserer programmeringsinnholdet i sannsynlighetstemaet i norske matematikkbøker for 9. trinn?* og *(2) På hvilke måter knytter lærebøkene programmering og sannsynlighet sammen?* i Kapittel 6. Jeg vil først presentere en oppsummering av funnene fra Kapittel 4.2. I Kapittel 5.1 vil mangelen på feilsøking diskuteres opp mot it-didaktikk før jeg videre i Kapittel 5.2 diskuterer om programmering er et egnet verktøy innenfor sannsynlighet ved hjelp av sannsynlighetsdidaktikk. Til slutt vil jeg diskutere mine funn opp mot resultatene til Bråting og Kilhamn (2021) og vurdere nyttigheten av deres analyseverktøy i denne studien.

Analyseenhetene har blitt analysert gjennom tre delanalyser som resulterte i flere funn. I delanalyse 1 ble det trukket frem to funn knyttet til handlinger: mangelen på feilsøking i analyseenhetene og fordelingen av handlingene *a) Følge en prosedyre* og *d) Forme og skape* i de to lærebøkene. Videre i delanalyse 2 om begreper ble det observert at 29 av 30 (97%) analyseenheter inneholdt sannsynlighetsbegreper og en forskjell i antall programmeringsbegreper funnet i de to lærebøkene, der Matemagisk inneholdt 53 programmeringsbegreper totalt kontra 28 identifisert i Maximum. Igjen observeres det en mangel på feilsøking innenfor programmeringsbegreper. Til slutt i delanalyse 3 om tilknytningen til sannsynlighet ble det funnet få enheter som bygde matematiske sammenhenger innenfor forskjellige sannsynlighetsbegreper, samt en forskjell i antall analyseenheter med kategorien *1. Elevenes tankegang og intuisjon* i de to lærebøkene.

5.1 Mangel på feilsøking

Feilsøking er kjent for å være en stor del av det daglige arbeidet for programvareutviklere (Perscheid et al, 2016). Likevel er Funn 1 og Funn 4 om mangel på feilsøking i lærebøkene i denne studien ikke et nytt fenomen, da Bråting og Kilhamn (2021) også observerte en mangel på feilsøking i svenske lærebøker på barneskolen. Denne likheten mellom de norske og svenske lærebøkene blir diskutert i Kapittel 5.3. I tillegg er det tidligere blitt opplevd at introduksjonslærebøker til programmering vier lite plass til feilsøking (McCauley, 2008).

Feilsøkings plass i læreplanen

En mulig grunn til at denne ferdigheten mangler fra lærebøkene kan være den spesifikke ordlyden i læreplanmålet som eksplisitt ber om at elevene skal «simulere utfall i tilfeldige forsøk og beregne sannsynligheten for at noe skal inntreffe, ved å bruke programmering». Jeg mener man kan svare på dette læreplanmålet uten bruk av feilsøking. Feilsøking er bare en liten del av CT-begrepet, som beskrevet i Kapittel 2.1, og er en ferdighet og tankegang elevene skal tilegne seg gjennom hele deres skolegang. Det kan da bli satt av tid til feilsøking i matematikkfaget på et annet trinn og i et annet tema enn i sannsynlighet på 9. trinn.

Siden fokuset i læreplanen er på CT og ikke programmering, vil det legges opp til at elevene lærer mer generelle anvendelige ferdigheter som å gi klare instruksjoner og problemløsning (Sentance et al., 2018). Da programmering er en del av CT, vil feilsøking være et nødvendig moment å inkludere på et overordnet nivå (Sentance et al., 2018). Om man går tilbake til Utdanningsdirektoratet sin definisjon av CT i Kapittel 2.2.3 ser vi at en av arbeidsmåtene som heves frem er feilsøking, der en del av algoritmisk tenkning er å oppdage og rette feil. I tillegg vil det å feilsøke føre til at elevene mulig er innom andre arbeidsmåter vist på Figur 2.3, som å *fikle* og *holde ut* da elevene kan møte på mer komplekse feil som krever eksperimentering og å fortsette og prøve igjen. Flere av nøkkelbegrepene kan elevene også få kjennskap til ved å feilsøke, avhengig av kompleksiteten til feilene. De må ha god kjennskap til *algoritmer* og *mønstre* i en kode, samt klare å fjerne unødvendige detaljer og gjøre vurderinger underveis om feilen er rettet opp. Dermed er feilsøking en ferdighet elevene kan forbedre gjennom å utvikle sin algoritmiske tenkning.

Algoritmisk tenkning blir spesifikt nevnt i kjerneelementet *utforskning og problemløsning*, som beskrevet i Kapittel 2.1, og viser at Utdanningsdirektoratet ser på CT som en problemløsningsmetode, der testing og feilsøking ses på som fundamentale momenter innenfor problemløsning (Sentance et al., 2018). Dette kjerneelementet finner vi i matematikkfaget. Selv om feilsøking er en metode rettet mot programmerere, er dette en måte å utvikle elevenes problemløsningsevner, og dermed deres CT-tankegang, der de får en annen vinkling på hvordan de kan løse utfordringer de støter på enn de tidligere har fått i den tradisjonelle undervisningen. Samtidig må man se på hva Utdanningsdirektoratet ønsker at elevene skal lære i løpet av grunnskolen. Hvis det store bildet er at de skal ha en oversikt over hele bredden til hva programmering kan oppnå, så er det viktig at de får erfare alle nøkkelbegrepene og praksisene innenfor algoritmisk tenkning. De lærer da også at programmering ikke er et spill eller programvare som skal læres, men et verktøy som krever ferdigheter for å bruke det, og denne prosessen inkluderer feilsøking (Sentance et al., 2018). Selvfølgelig skal ikke alle elever bli programmerere, men det er viktig at de har muligheten til å forstå verdenen de vokser opp i, der digitale systemer i økende grad er en del av nesten alt vi gjør (Sentance et al., 2018).

Feilsøkings plass i matematikkfaget

Feilsøking er et konsept som er programmeringsspesifikt og kan bli tatt mindre hensyn til grunnet programmeringens innføring inn i det allerede innholdsrike matematikkfaget, der matematikken vil være i fokus. Som Bråting og Kilhamn (2021) også poengterte virker det som at fokuset er skiftet bort fra å lære det som er nødvendig for programmering til å sørge for at det nye innholdet fortsatt er relevant for det som har vært i matematikkfaget tidligere.

Samtidig vil matematikkfaget være et naturlig sted å lære elevene om feilsøking siden det er i dette faget det forventes at det settes av tid til å tilegne elevene ferdigheter innenfor algoritmisk tenkning. I Norge er algoritmisk tenkning og programmering introdusert inn i de eksisterende fagene matematikk, naturfag, musikk og kunst og håndverk (Bocconi et al., 2022, s. 49). Ut fra kompetansemålene for disse fagene vil man se flest mål som omhandler programmering i matematikkfaget. Med tanke på feilsøking sier et kompetansemål for matematikk etter 8. trinn at eleven skal "utforske hvordan algoritmer kan skapes, testes og forbedres ved hjelp av programmering", som

er det nærmeste kompetansemålet jeg finner som indirekte nevner feilsøking ved at det skal testes og forbedres (Kunnskapsdepartementet, 2020a). Om man vil kan et kompetansemål for kunst og håndverk etter 7. trinn kunne ta for seg feilsøking, da det sier at eleven skal kunne «bruke ulike strategier for idéutvikling og problemløsing», men målet vil trolig bli svart på uten denne ferdigheten (Kunnskapsdepartementet, 2020b). Ut fra kompetansemålene vil det dermed være trolig at matematikk er det faget elevene mulig kan få undervisning om feilsøking.

Muligheter for feilsøkingsproblemer

Bråting og Kilhamn (2021) nevner at de ser et potensiale i å kunne utvide bredden i programmeringsoppgaver til å inneholde flere av handlingene *f) Forestille seg* og *c) Feilsøke* som kan koble matematikk og programmering enda mer sammen, som ved å lære matematikk ved å teste og feilsøke. Derfor burde feilsøkingsstrategier bli innlemmet i matematikkfaget og bli eksplisitt undervist for å gi elevene ulike metoder å gå frem på når de møter utfordrende problemer (Sentance et al., 2018). Det er også undersøkt at det å få undervisning i feilsøking vil utvikle elevenes ferdigheter i å diagnostisere og fjerne feil fra koder (Chmiel & Loui, 2004). Spesielt ble det funnet at elever som fullførte valgfrie feilsøkingsøvelser brukte betydelig mindre tid på å feilsøke programmene sine enn kontrollgruppen av elever som ikke prøvde seg på disse øvelsene. Det viser at det å ha tilgang til og gjøre oppgaver om feilsøking vil utvikle elevenes evner innenfor dette, og dermed også algoritmisk tenkning. Det burde derfor bli laget oppgaver som kan teste elevenes evner innenfor feilsøking. I tillegg er feilsøking en måte å få en større forståelse for hvordan en kode kjører og «reagerer» på ulike betingelser, som kan gjøre en til en bedre programmerer.

Mangelen på feilsøking kan dermed forekomme av ordleggingen i kompetansemålene som i sannsynlighet henviser til programmering som et verktøy og ikke en egen kunnskap, og hensikten med å innføre programmering og algoritmisk tenkning i skolen. Samtidig må det igjen poengteres at jeg kun har undersøkt sannsynlighetstemaet i lærebøker for 9. trinn og det er mulig at det eksisterer feilsøking i andre temaer eller trinn der det kan ha blitt ansett som mer hensiktsmessig og plass i undervisningen til konseptet.

5.2 Hva tilfører programmering til oppgavene i sannsynlighet?

Gjennom delanalysen om begreper i Kapittel 4.1.2 ser man ut fra det høye antallet av begrepet *simulere/simulering* at programmering er henholdsvis brukt som et simuleringsverktøy for å utføre ulike random device eksperimenter såpass mange ganger at det er urealistisk å gjennomføre fysisk i klasserommet. Det er mulig å utføre, som matematikeren John Edmund Kerrich som utførte et myntkast 10 000 ganger for å demonstrere Store talls lov og hvordan gjennomsnittet av resultatene fra disse kastene nærmet seg den teoretiske verdien. Det skal derimot bemerkes at Kerrich utførte dette i et Nazi-okkupert Danmark på 1940-tallet og hadde trolig ikke så mye annet å finne på. Begrepet *simulere/simulering* ble spesifikt funnet i over 50% av analyseenhetene (17 av 30 totalt), som førte til innlemmelsen av dette begrepet i programmeringsbegreper. Jeg beskrev i Kapittel 2.5 om Biehler (2019) sitt skille av funksjonene ved simulering, der den første funksjonen er å bruke simulering som en visualisering og gjøre sannsynlighet

og sjanse til en større del av elevenes hverdag og gi dem en større forståelse og intuisjon innenfor temaet. Den andre pedagogiske funksjonen simulering kunne ha var at det var en del av prosessen med modellering og problemløsning. Den første funksjonen stemmer overens med funnene av kategoriene *1. Elevenes tankegang og intuisjon* og *2. Knytte virkelighet og teori sammen* i datamaterialet i delanalyse 3. Man kan dermed tenke seg at programmering har blitt gitt denne funksjonen i lærebøkene. Derimot krever den andre funksjonen en viss programmeringskompetanse hos elevene for at de skal kunne bruke dette som et hensiktsmessig verktøy slik at de kan konstruere egne modeller og simuleringer.

Fra Funn 2 ble det observert at Maximum inneholdt mer av handlingen *d) Forme og skape*, samt at man i Funn 4 så en mangel på eksplisitte begreper i denne læreboka. Det virker dermed som at Matemagisk og Maximum har en ulik forventning til kompetansen elevene har innenfor programmering, der Maximum gir elevene mer åpne oppgaver med mindre informasjon mens Matemagisk gir elevene mer hjelp til å løse oppgavene ved å dekomponere problemet for dem og spesifikt nevne programmeringsbegreper. Videre kunne man sett på om det er de mer lukkede og rettede oppgavene som man finner i Matemagisk eller de mer åpne oppgavene i Maximum, som er best egnet for programmeringsoppgaver i en matematisk kontekst. Eller om man burde ha en blanding av disse forskjellige metodene for å strukturere en oppgave som kan gi elevene best mulig læringspotensial. Åpne oppgaver kan derimot føre til et krav om en viss mengde programmeringskompetanse hos lærere for å kunne tilrettelegge for god undervisning og hjelpe elevene underveis. I min tidligere studie fra emnet RFEL3100 på NTNU fortalte lærere meg gjennom intervjuer at de opplevde at de hadde for lav kompetanse innenfor programmering til å gi elevene god undervisning (Hollås, 2023). De ønsket sterkt mer opplæring og ressurser innenfor programmering da ingen lærere er komfortable med å undervise et tema som de kanskje bare kan 1 eller 2 leksjoner mer enn elevene. Stigberg og Stigberg (2020) fant de samme funnene i sin studie der de undersøkte hvordan programmering ble undervist i matematikkfaget i Sverige. De opplevde at lærere følte en mangel på tilstrekkelig kunnskap om undervisning i programmering. Lærerne pekte på at de ønsket solide ressurser som kunne hjelpe dem med å tilegne seg kunnskap om programmering og undervisning i temaet. Det virker dermed som at man i praksis møter en utfordring i læreres kompetanse i undervisning med programmering. Dette betyr at lærerne får et mindre spillerom i undervisningen og de vil finne det vanskeligere å vurdere programmeringsinnholdet i ulike ressurser for undervisning, som for eksempel i lærebøkene.

Samtlige av analyseenhetene inneholdt også en tilknytning til virkeligheten ved bruk av random device eller liknende, der *2. Knytte virkelighet og teori sammen* ofte ble kategorisert. Dette ble mye brukt for å få frem store talls lov og nyttigheten av å bruke et digitalt verktøy når man skal gjennomføre et eksperiment et visst antall ganger. Som Batanero et al. (2005) skrev, så kan man ved hjelp av programmering utføre simuleringer i skolen, der elevene kan modellere og utforske tilfeldige fenomener og utvikle en intuisjon og forståelse innenfor sannsynlighet. Dette samsvarer også med læreplanmålet for 9. trinn der elevene skal «simulere utfall i tilfeldige forsøk og beregne sannsynligheten for at noe skal inntreffe, ved å bruke programmering». Ser man over datamaterialet og hvilke analyseenheter som bruker programmering for å simulere ulike utfall, vil dette bli funnet i 13 av 16 (81%) enheter fra Matemagisk og i alle 14 (100%) enheter fra Maximum. Man kan dermed si at nesten alle analyseenhetene som inneholder programmering i sannsynlighetskapitlene i lærebøkene svarer spesifikt på dette læreplanmålet. Algoritmisk tenkning og spesifikke simuleringer kan hjelpe elevenes

forståelse av bruksområdet til modeller for å beskrive virkeligheten (Serpe & Frassia, 2017). Dette gjenkjenner vi som kjerneelementene *modellering og anvendelser* og *representasjon og kommunikasjon* ved å gi elevene tilgang til ulike representasjoner av sannsynlighetsbegreper, sammenhenger og problemer. I tillegg vil dette presentere teorien på en klarere måte for elevene, som videre gir oss kjerneelementene *abstraksjon og generalisering* og *matematiske kunnskapsområder*. Utdanningsdirektoratet anser kjerneelementene som det viktigste faglige innholdet i matematikkfaget, som betyr at simulering er et verktøy som kan hjelpe med å få denne kunnskapen frem.

I Funn 5 kom det frem at begge lærebøkene inneholdt lite matematiske sammenhenger innenfor sannsynlighet. Det skal igjen påpekes at jeg i denne studien bare har analysert enheter som inneholder programmering, så det er mulighet for at sammenhengene i sannsynlighetstemaet blir utforsket i de programmeringsfrie oppgavene der matematikken tar all plassen i problemene.

Samtidig poengterte Batanero et al. (2005) at siden simulering bare gir problemløsningen, og ikke grunnen til at løsningen er gyldig, har den ingen forklaringskraft og burde støttes opp av den mer tradisjonelle undervisningen av de formelle reglene innenfor sannsynlighet. Det vil derfor være interessant å analysere hele sannsynlighetskapittelet med alle enheter, både om de inneholder programmering eller ikke. Da kan man se på helheten og om oppgavene med og uten programmering klarer å bygge på hverandre og gi elevene en forklaring på hvorfor simuleringen de utførte gir et gyldig svar.

5.3 Vurdering opp mot Bråting og Kilhamn (2021)

Under studien har jeg videreutviklet og delvis tilpasset Bråting og Kilhamn (2021) sitt rammeverk for handlinger og begreper. Avslutningsvis vil jeg diskutere mine funn opp mot det som ble funnet i de svenske lærebøkene for barneskolen og vurdere nyttigheten av rammeverket i min studie. Jeg vil først ta for meg handlinger, før jeg går videre til begreper. Til slutt vil jeg diskutere koblingen mellom programmering og matematikk.

Handlinger

Av handlinger har både de norske og svenske lærebøkene i studiene flest av handlingen *a) Følge en prosedyre* og deretter *d) Forme og skape*, samt lite eller ingen funn av *c) Feilsøke*. Som beskrevet i Kapittel 2.2.5 har det vært hensiktsmessig i min studie å bruke de samme handlingene som Bråting og Kilhamn (2021), der jeg senere i analyseprosessen ikke trengte å utvide eller ta bort noen handlinger.

Fra resultatene fra Bråting og Kilhamn (2021) og min studie ser man en trend i at lærebøkene analysert fokuserer på noen handlinger, men utelater andre som er fundamentale innenfor læring av programmering som *c) Feilsøke*. Både Norge og Sverige har implementert programmering inn i eksisterende fag, som i Sverige er matematikk, teknologi og samfunnsfag (Bocconi et al., 2022, s. 48). I tillegg blir CT-konsepter og programmering hovedsakelig behandlet i matematikk i begge landene. Bocconi et al. (2022) fant i sin rapport at hovedmotivasjonen for å introdusere algoritmisk tenkning i de fleste land er for å fremme *21st Century skills*, som blir forstått som essensielle for å kunne delta aktivt i den digitale verden. Siden problemløsning og teknologiske

ferdigheter er en del av 21st Century skills, kan det forklare at hovedtyngden for læring av CT blir lagt i matematikkfaget, da programmering er den mest representerte CT-praksisen i både Sverige og Norge (Vinnervik & Bungum, 2022, s. 397). Begge lands kompetansemål for matematikkfaget knytter programmering til å være et digitalt verktøy for å utforske matematiske problemer istedenfor spesifikke programmeringsproblemer (Vinnervik & Bungum, 2022, s. 390). På samme måte som i den norske læreplanen, er dermed målet med programmering i Sverige at det skal brukes som et verktøy for å utforske problemer og matematiske konsepter, utføre kalkulasjoner og tolke data. Læring av programmering blir dermed sidestilt for å passe på at det nye innholdet fortsatt er relevant for å lære tradisjonell matematikk, siden man henviser til programmering som et verktøy for matematikk i kompetansemålene i begge land.

Videre er det en forskjell i funnene av handlingene *e) Forklare* og *f) Forestille seg*, der jeg i min studie har funnet disse handlingene i 24 av 30 (80%) analyseenheter kontra i rundt 35 av 390 (9%) analyseenheter hos Bråting og Kilhamn (2021). Jeg vil anta at mye av denne forskjellen ligger i at Bråting og Kilhamn (2021) har analysert lærebøker for barneskolen, mens jeg har sett på lærebøker for ungdomsskolen, spesifikt 9. trinn. I løpet av årene på grunnskolen forventes det mer og mer av elevene da de lærer og tilegner seg mer kunnskap. Dette kan også forklare forskjellige funn innenfor programmeringsbegreper, samt at jeg har analysert lærebøker som går ut fra tekstprogrammering mens Bråting og Kilhamn (2021) har forholdt seg til lærebøker som inneholder analog og blokkprogrammering.

Begreper

Bråting og Kilhamn (2021) fant at begrepet *algoritme* ble brukt på mellomtrinnet, mens man på barnetrinnet bare forholdt seg til *stegvise instruksjoner*. Jeg fant liknende at *algoritme* ble mye brukt i lærebøkene for 9. trinn. Man introduserer mer komplekse begreper jo eldre man blir i skoleløpet. På samme måte var begrepet *program*, som ikke finnes i Bråting og Kilhamn (2021), begrepet som ble mest brukt i min studie. Dette kan forklares av hoppet til ungdomstrinnet og bruken av tekstprogrammering, der *program* er et mer passende og brukt begrep innenfor denne grenen av programmering.

Mens Bråting og Kilhamn (2021) har sett på forskjeller i begrepsbruk på barnetrinnet og mellomtrinnet, har jeg sett på lærebøker fra to ulike forlag, der jeg fant at Maximum inneholdt generelt færre begreper grunnet sine åpne oppgaver. Dermed er det ikke like interessant å diskutere funnene av begrepene *betingelse*, *regel*, og *løkker/iterasjon/repetisjon*, der disse begrepene enten økte eller minket i bruk mellom barnetrinnet og mellomtrinnet i funnene til Bråting og Kilhamn (2021). Jeg kan ikke diskutere dette på noen måte med hvordan begrepsbruken arter seg på ungdomsskolen grunnet så stor forskjell mellom lærebøkene analysert. I tillegg var dette begreper originalt fra funnene til Bråting og Kilhamn (2021), som betyr at jeg før analyseprosessen ikke hadde en formening om disse ble brukt i mitt datamaterialet eller ikke. Men, siden disse begrepene var like relevante for tekstprogrammering som blokkprogrammering, anså jeg de som hensiktsmessige å gå ut fra. Som beskrevet i Kapittel 2.2.4 var jeg nødt til å legge til noen begreper som da ikke finnes i Bråting og Kilhamn (2021) sin studie og dermed ikke er relevant å diskutere.

For matematiske begreper er det heller ingen grunn til å diskutere funnene grunnet forskjellig vinkling på studiene. Sverige har lagt programmeringsfokuset innenfor algebra

som gjorde de matematiske begrepene til Bråting og Kilhamn (2021) mindre passende i min studie med fokuset på sannsynlighet (Vinnervik & Bungum, 2022, s. 389). Jeg kunne dermed ikke gå ut fra de matematiske begrepene funnet av Bråting og Kilhamn (2021) da dette ble for bredt, som førte til mitt først induktive arbeid for å lage egne underkategorier for deretter å deduktiv lete etter disse i mitt datamateriale.

Kobling av matematikk og programmering

I Bråting og Kilhamn (2021) sin studie ser de først på handlinger og begreper, før de ser på kombinasjonen av disse og ser på hvordan matematikken og programmeringen kobles sammen i lærebøkene. Grunnet min innsnevring mot sannsynlighetstemaet valgte jeg å lage en egen modell ved bruk av Castro (1998) og Wilensky (1995) som kan si noe om sannsynlighetsinnholdet i datamaterialet. Denne begrensingen i min studie førte derimot til at jeg ikke kan si om funnene mine er spesielle for sannsynlighetsdelene siden jeg ikke har sett på de andre temaene i bøkene.

Under analyseprosessen har rammeverket til Bråting og Kilhamn (2021) vist seg å være et relevant og passende analyseverktøy for programmeringsinnholdet innenfor sannsynlighet i lærebøkene for 9. trinn. Ved å analysere etter handlinger og begreper fikk jeg en god oversikt over hvordan de forskjellige lærebøkene har innlemmet programmering inn i kapitlene og kan overordnet fortelle om dette innholdet. Som nevnt, utførte jeg analysen etter matematiske begreper på en annen måte samt at det måtte legges til noen programmeringsbegreper. Dette kommer trolig av forskjellen i studiens aldersgruppe som lærebøkene er ment for, samt hvilke programmeringsgrener som har blitt benyttet i lærebøkene.

5.4 Studiens begrensninger

I denne studien har mitt fokus vært på hvordan programmering har blitt innført inn i sannsynlighetstemaet i lærebøkene for 9. trinn, ved å analysere de to lærebøkene Matemagisk og Maximum. I mine funn har jeg påpekt mangel på feilsøking og ulik struktur og intensjon i enhetene i de to lærebøkene. Det fremvises bare fakta og ingen vurdering av datamaterialet, men funnene kan anses som en indirekte vurdering av kvaliteten av innholdet i lærebøkene, selv om dette ikke var intensjonen. Samtidig er dette vanskelig å unngå grunnet studiens metode av tekstbokanalyse som spesifikt ser på innholdet i lærebøkene, samt de teoretiske rammeverkene som kan gi et innsyn i hva som karakteriserer datamaterialet.

En svakhet i min studie er det begrensede datamaterialet grunnet fokuset på sannsynlighetstemaet på 9. trinn. Jeg har kun undersøkt et kapittel i begge lærebøkene og kan ikke ta stilling til hva som karakteriserer innholdet i resten av lærebøkene eller hvordan temaene i boka helhetlig benytter seg av programmering. På den andre siden førte denne innsnevringen mot sannsynlighet til at jeg kunne se nøyaktig på sannsynlighetsdidaktikk som ikke hadde vært like relevant i en generell studie. I tillegg har jeg kun undersøkt enheter som inneholder programmering i sannsynlighetstemaet, som gjør at datamaterialet mitt ikke inneholdt de tradisjonelle sannsynlighetsoppgavene. Jeg kan ikke si noe om de enhetene med og uten programmering knyttes sammen på noen måte. Flere forlag har flere ressurser for programmering på sine nettsider da disse enklere kan oppdateres og utvides, men grunnet at jeg kun har sett på de fysiske

lærebøkene kan jeg kun fortelle om programmeringsinnholdet i disse. Skolene kjøper først inn de fysiske lærebøkene og har ikke nødvendigvis tilgang til de digitale ressursene før etter kjøpet. Derfor har jeg ansett det som hensiktsmessig å bare vurdere de fysiske lærebøkene grunnet at det er det skolene vurderer før de kjøper inn fra et forlag.

En annen svakhet med studien er det store antallet delanalyser med koder og kategorier å forholde seg til under analyseprosessen. Dette kan føre til at noen av kodene har blitt brukt ulikt. Jeg har i tillegg vært alene i analyseprosessen, kontra for eksempel Bråting og Kilhamn (2021) som i sin studie kunne diskutere sine tolkninger med hverandre, som kan gjøre at analysen blir farget av mine oppfatninger av læreverkene. Jeg har allikevel prøvd å redusere denne svakheten ved å beskrive alle delanalysene i detalj og har eksplisitt forklart hvordan kodene har blitt brukt, samt å diskutere min koding med en medstudent eller med min veileder hvis jeg anså noe som uklart. Dette førte til en mer detaljert gjennomgang av kriteriene i de ulike rammeverkene og hjalp meg i analyseprosessen da jeg ble mer trygg på mine valg og tolkninger.

6 Konklusjon

Som følge av LK20 ble programmering og algoritmisk tenkning innført i matematikkfaget, der programmering spesifikt blir nevnt som et verktøy i kompetansemålet for 9. trinn som omhandler sannsynlighet. Denne studien hadde som formål å få et innblikk inn i programmeringsinnholdet i sannsynlighetstemaet i lærebøker for 9. trinn. For å undersøke dette, valgte jeg å stille spørsmålene:

1) Hva karakteriserer programmeringsinnholdet i sannsynlighetstemaet i norske matematikkbøker for 9. trinn?
2) På hvilke måter knytter programmeringsoppgavene programmering og sannsynlighet sammen?

Studien er en kvalitativ innholdsanalyse, hvor jeg har gjennomført en analyse av datamateriale fra lærebøkene Matemagisk og Maximum ved bruk av et videreutviklet rammeverk ut fra Bråting og Kilhamn (2021) sitt analyseverktøy i kombinasjon med et egenutviklet rammeverk for læring av sannsynlighet ut fra sannsynlighetsdidaktikk.

Forskningsspørsmål 1

Programmeringsinnholdet i sannsynlighetstemaet i norske matematikkbøker for 9. trinn karakteriseres av at eleven skal enten kopiere, utvide eller lage en egen kode ved bruk av tekstprogrammering i Python, samt en mangel på problemer med feilsøking. Generelt er handlingene *a) Følge en prosedyre* og *d) Forme og skape* de mest identifiserte i datamaterialet. Fra analyseprosessen er det klassifisert flere handlinger og begreper i Matemagisk enn i Maximum og man ser en ulik fordeling av handlinger og begreper i de to analyserte lærebøkene grunnet forskjell i tilnærming i temaet. Matemagisk inneholder flere deloppgaver med handlingene *a) Følge en prosedyre*, *e) Forklare* og *f) Forestille seg*, kontra Maximum sine mer åpne oppgaver med handling *d) Forme og skape*. I begge lærebøker ble det derimot ikke gjort noen funn av handling *c) Feilsøke* eller noen programmeringsbegreper som omhandler feilsøking.

Forskningsspørsmål 2

Dataene viser at programmeringsoppgavene knytter programmering og sannsynlighet sammen ved å benytte programmering hovedsakelig som et verktøy for simulering og beregning, der kunnskaper og ferdigheter innenfor sannsynlighet holdes sentralt i lærebøkene. Ut fra den egenutviklede modellen for læring av sannsynlighet ble det observert mest av kategoriene *2. Knytte virkelighet og teori sammen* og *4. Oppsett*, siden programmering blir brukt som et simuleringsverktøy for ulike virkelighetsnære problem og random device eksperimenter. Ut fra analysen ble det observert en ulik fordeling av kategori *1. Elevenes tankegang og intuisjon* der Matemagisk inneholdt flere enheter med denne kategorien med flere deloppgaver som ba elevene tenke, forklare og reflektere over resultatene de fikk fra utført simulering. I begge lærebøker ble det identifisert få funn av kategori *3. Matematiske sammenhenger* hvor dataene besto hovedsakelig av simuleringsproblemer om store talls lov uten kobling til andre momenter og konsepter i sannsynlighet.

Som nevnt i Kapittel 5.2 vil videre forskning kunne være å analysere hele sannsynlighetskapittelet med alle enheter for å se på helheten, og om oppgavene med og uten programmering klarer å bygge på hverandre og gi elevene en forklaring på hvorfor simuleringen de utførte gir et gyldig svar. Jeg vil også foreslå å kartlegge de digitale ressursene om programmering til lærebøkene som en forlengelse av denne studien. De digitale ressursene vil trolig bli mer og mer brukt i skolen når hverdagen blir mer digitalisert, og vil derfor kunne ha en stor innflytelse på hva og hvordan ulike temaer blir undervist i matematikkfaget. I denne studien har jeg, som beskrevet i Kapittel 1.2, fokusert på sannsynlighet da et av kompetansemålene for sannsynlighet eksplisitt nevner at programmering skal brukes. Mitt fokus på sannsynlighet lot meg analysere datamaterialet med den egenutviklede modellen ut fra sannsynlighetsdidaktikk, som ville vært lite hensiktsmessig å bruke i analyse av andre temaer. Ved å være bevisst på lærebøkenes styrker, mangler og forskjeller i innhold og struktur, som denne studien har belyst i sannsynlighetskapitlene for 9. trinn, kan lærere få et innblikk inn i lærebøkene og vurdere og tilpasse hva som passer til deres undervisning. Bråting og Kilhamn (2021) sitt rammeverk over handlinger og begreper kan benyttes og videreutvikles av lærere som vil undersøke programmeringsinnholdet i andre temaer.

Referanser

Batanero, C., Biehler, R., Maxara, C., Engel, J. & Vogel, M. (2005). *Using simulation to bridge teachers' content and pedagogical knowledge in probability.* [Paperpresentasjon]. 15th ICMI Study Conference: The Professional education and development of teachers of mathematics, São Paulo, Brazil. https://www.researchgate.net/publication/282281200_Using_simulation_to_bridge_teachers_content_and_pedagogical_knowledge_in_probability

Benton, L., Hoyles, C., Kalas, I., & Noss, R. (2016, februar). *Building mathematical knowledge with programming: insights from the ScratchMaths project*. [Paperpresentasjon]. Constructionism 2016, Bangkok, Thailand. https://www.researchgate.net/publication/295912532_Building_mathematical_knowledge_with_programming_insights_from_the_ScratchMaths_project

Benton, L., Hoyles, C., Kalas, I., & Noss, R. (2017). Bridging primary programming and mathematics: Some findings of design research in England. *Digital Experiences in Mathematics Education, 3*, 115-138. https://doi.org/10.1007/s40751-017-0028-x

Biehler, R. (2019, februar). *Software for learning and for doing statistics and probability - Looking back and looking forward from a personal perspective.* [Paperpresentasjon]. Congreso International Virtual de Educación Estadística, Granada. https://www.researchgate.net/publication/331284631_Software_for_learning_and_for_doing_statistics_and_probability-Looking_back_and_looking_forward_from_a_personal_perspective

Bocconi, S., Chioccariello, A. & Earp, J. (2018). *The Nordic approach to introducing computational thinking and programming in compulsory education*. Nordic@BETT2018 Steering Group. http://dx.doi.org/10.17471/54007

Bocconi, S., Chioccariello, A., Kampylis, P., Dagienė, V., Wastiau, P., Engelhardt, K., Earp, J., Horvath, M.A., Jasutė, E., Malagoli, C., Masiulionytė-Dagienė, V. & Stupurienė, G. (2022). *Reviewing Computational Thinking in Compulsory Education*. I A. Inamorato dos Santos, R. Cachia, N. Giannoutsou & Y. Punie (Red.). Publications Office of the European Union, Luxembourg. https://publications.jrc.ec.europa.eu/repository/handle/JRC128347

Brennan, K., & Resnick, M. (2012, 13.-17. april). *New frameworks for studying and assessing the development of computational thinking.* [Paperpresentasjon]. Proceedings of the 2012 annual meeting of the American educational research association, Vancouver, Canada. http://scratched.gse.harvard.edu/ct/files/AERA2012.pdf

Bråting, K., & Kilhamn, C. (2021). The integration of programming in Swedish school mathematics: Investigating elementary mathematics textbooks. Scandinavian Journal of Educational Research. *Scandinavian Journal of Educational Research, 66*(4), 594-609. https://doi.org/10.1080/00313831.2021.1897879

Bybee, R.W., Taylor, J., Gardner, A., Scotter, P., Carlson, J., Westbrook, A. & Landes, N. (2006). The BSCS 5E instructional model: Origins and effectiveness. Office of Science Education National Institutes of Health. https://www.researchgate.net/publication/281412517_The_BSCS_5E_instructional_model_Origins_and_effectiveness

Castro, C. S. (1998). Teaching Probability for Conceptual Change La Enseñanza De La Probabilidad Por Cambio Conceptual. *Educational Studies in Mathematics, 35*, 233-254. https://doi.org/10.1023/A:1003182219483

Cohen, L., Manion, L., & Morrison, K. (2018). *Research Methods in Education* (8. utg.). Routledge.

FN-sambandet. (2024, 30. januar). *God utdanning*. Hentet 12. mai 2024 fra https://fn.no/om-fn/fns-baerekraftsmaal/god-utdanning

Grønmo, S. (2023, 15. april). Utvalg. I *Store norske leksikon*. https://snl.no/utvalg

Hjardar, E. & Pedersen J. (2021). *Matematikk 9*. Cappelen Damm.

Hollås, M. T. (2023). *Matematikklæreres opplevelse av programmering i undervisningen*. [Upublisert rapport fra en studie i emnet RFEL3100]. Norges teknisk-naturvitenskapelige universitet, Trondheim.

Kivle, S. O. (u.å.). *Blokk-koding*. skolekoding.no. Hentet 19. november 2023 fra https://skolekoding.no/blokk/

Kongsnes, A. L. & Wallace, A. K. (2022). *Matemagisk 9*. Aschehoug.

Krippendorff, K. (2013). *Content analysis. An Introduction to Its Methodology* (3. utg.). Sage.

Kuckartz, U. (2019). Qualitative text analysis: A systematic approach. I G. Kaiser & N. Presmeg (Red.). *Compendium for Early Career Researchers in Mathematics Education* (s. 181-197) ICME-13 Monographs, Springer.

Kunnskapsdepartementet. (2018, 26. juni). *Fornyer innholdet i skolen*. Hentet 15. mai 2024, fra https://www.regjeringen.no/no/aktuelt/fornyer-innholdet-i-skolen/id2606028/

Kunnskapsdepartementet. (2020a). *Læreplan i matematikk 1-10 (MAT01-05)*. Fastsatt som forskrift. Læreplanverket for Kunnskapsløftet 2020. https://www.udir.no/lk20/mat01-05

Kunnskapsdepartementet. (2020b). *Læreplan i kunst og håndverk (KHV01-02)*. Fastsatt som forskrift. Læreplanverket for Kunnskapsløftet 2020. https://www.udir.no/lk20/khv01-02

Lær Kidsa Koding (u.å.). *Python*. Lær Kidsa Koding. Hentet 19. november 2023 fra https://www.kidsakoder.no/ufaq/python/

Neraal, A. (2024, 22. januar). Norges 50 største forlag. *Bok365.* https://bok365.no/artikkel/norges-50-storste-forlag/

NESH. (2021). Forskningsetiske retningslinjer for samfunnsvitenskap og humaniora (5. utg.). De nasjonale forskningsetiske komiteene. https://www.forskningsetikk.no/retningslinjer/hum-sam/forskningsetiske-retningslinjerfor-samfunnsvitenskap-og-humaniora/

Rezat, S. & Strässer, R. (2017). Methodological issues and challenges in research on mathematics textbooks. I B. Grevholm (Red.), *Mathematics textbooks, their content, use and influences. Research in Nordic and Baltic countries* (s. 495-514). Cappelen Damm Akademisk.

Scratch. (u.å.). *Om Scratch*. Hentet 20. april 2024 fra https://scratch.mit.edu/about

Sentance, S., Barendsen, E. & Schulte, C. (2018). *Computer Science Education: Perspectives on Teaching and Learning in School*. Bloomsbury Academic.

Serpe, A. & Frassia, M. G. (2017, februar). *Technology will solve student's probability misconceptions: Integrating simulation, algorithms and programming*. [Paperpresentasjon]. CERME 10, Dublin, Irland. https://www.semanticscholar.org/paper/Technology-will-solve-student's-probability-and-Serpe-Frassia/d75df3b1dff0a2b62054b8bec020490655402859?utm_source=direct_link

Statped. (2021, 1. mars). *Programmering*. Hentet 24. april 2024 fra https://www.statped.no/laringsressurser/teknologitema/programmering-for-barn-med-saerskilte-behov/programmering/hva-er-programmering/

Stigberg, H. & Stigberg, S. (2020). Teaching programming and mathematics in practice: A case study from a Swedish primary school. *Policy Futures in Education, 18*(4), 483-496. https://doi.org/10.1177/1478210319894785

Tofteberg, G. N., Tangen, J., Bråthe, L. T., Stedøy, I. & Alseth, B. (2021). *Maximum 9* (2. utg.). Gyldendal.

Tjora, A. (2017). Kvalitative forskningsmetoder i praksis. (3. utg.). Oslo: Gyldendal Akademisk.

Utdanningsdirektoratet. (2019a, 27. mars). *Algoritmisk tenkning*. https://www.udir.no/kvalitet-og-kompetanse/digitalisering/algoritmisk-tenkning/

Utdanningsdirektoratet. (2019b, 18. november). *Hva er kjerneelementer?* https://www.udir.no/laring-og-trivsel/lareplanverket/stotte/hva-er-kjerneelementer/

Vihovde, E. H. (2023, 14. september) programmering - IT. I *Store norske leksikon*. https://snl.no/programmering_-_IT

Vinnervik, P. & Bungum, B. (2022). Computational thinking as part of compulsory education: How is it represented in Swedish and Norwegian curricula?. *Nordina*, *18*(3), 384-400. https://doi.org/10.5617/nordina.9296

Wilensky, U. (1995). Paradox, programming and learning probability: A case study in a connected mathematics framework. *Journal of mathematical behavior, 14*(2), 253-280. https://doi.org/10.1016/0732-3123(95)90010-1

Wing, J. M. (2006). Computational thinking. *Communications of the ACM, 49*(3), 33-35. http://dx.doi.org/10.1145/1118178.1118215

Vedlegg

Vedlegg 1: Eksempel på analyseenhet med handling *e) Forklare* fra Matemagisk

Vedlegg 2: Spesiell analyseenhet fra Matemagisk

Vedlegg 3: Spesiell analyseenhet fra Maximum

Vedlegg 1:

13B Store talls lov

- Bruke **programmering** til å **simulere** tilfeldige utfall.
- **Utforske** og kjenne til sammenhengen mellom relativ frekvens og sannsynlighet.
- Bruke **relativ frekvens** til å finne sannsynligheter.

På 8. trinn lærte vi at:	Eksempel
Vi kan bruke en **for-løkke** når vi vil gjenta kodelinjer et bestemt antall ganger.	`for tall in range(10):` `  print(tall)`

Vi kan bruke en **if-setning** for å utføre enkelte kodelinjer bare hvis en betingelse er sann.	`if alder < 18:`
	`  timelønn = 130`
	`else:`
	`  timelønn = 155`

Å SIMULERE TERNINGKAST

Vi kan kaste en terning 100 ganger, lage en frekvenstabell og finne den relative frekvensen for hvert av de seks utfallene. Hvis vi gjør dette, kan resultatet for eksempel bli slik:

Utfall	Antall ganger (frekvens)	Relativ frekvens
Ener	20	0,2
Toer	15	0,15
Treer	12	0,12
Firer	18	0,18
Femmer	8	0,08
Sekser	27	0,27
Sum	**100**	**1**

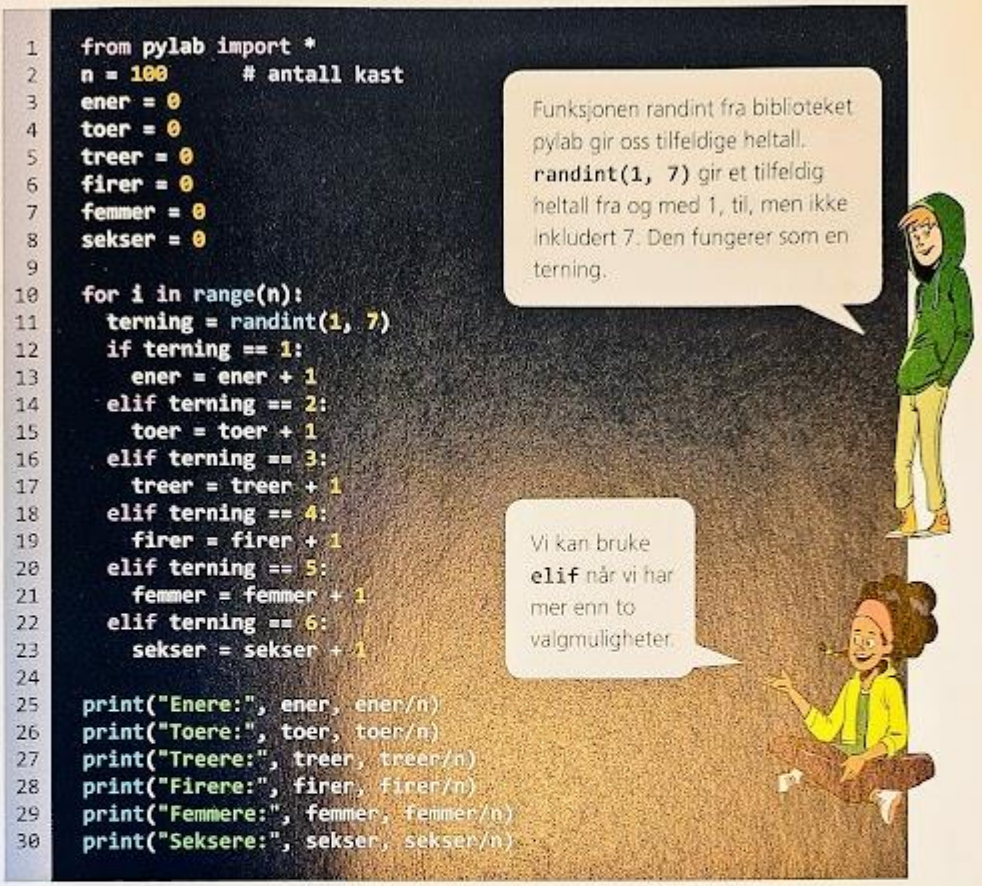
Vi kan lage et pythonprogram som simulerer 100 terningkast, teller hvor mange vi har av hvert utfall og regner ut relativ frekvens.

Algoritme

Steg 1 Importer biblioteket pylab og opprett en variabel **n** som står for antall kast.
Steg 2 Opprett en variabel for hvert utfall og gi variablene verdien 0.
Steg 3 Gjenta 100 ganger:
  Kast terningen.
  Øk variabelen som teller det utfallet vi fikk, med 1.
Steg 4 Skriv frekvens og relativ frekvens til skjermen.

Program i Python

```
1  from pylab import *
2  n = 100       # antall kast
3  ener = 0
4  toer = 0
5  treer = 0
6  firer = 0
7  femmer = 0
8  sekser = 0
9
10 for i in range(n):
11   terning = randint(1, 7)
12   if terning == 1:
13     ener = ener + 1
14   elif terning == 2:
15     toer = toer + 1
16   elif terning == 3:
17     treer = treer + 1
18   elif terning == 4:
19     firer = firer + 1
20   elif terning == 5:
21     femmer = femmer + 1
22   elif terning == 6:
23     sekser = sekser + 1
24
25 print("Enere:", ener, ener/n)
26 print("Toere:", toer, toer/n)
27 print("Treere:", treer, treer/n)
28 print("Firere:", firer, firer/n)
29 print("Femmere:", femmer, femmer/n)
30 print("Seksere:", sekser, sekser/n)
```



SNAKKE MATTE

Se på programmet ovenfor.

a Forklar hvilke linjer i programmet som hører til hvilket punkt i algoritmen.

b Forklar hvordan programmet finner hvilken variabel som skal økes med 1 etter at terningen er kastet.

c Forklar hvordan de relative frekvensene regnes ut.

d Hva forventer dere at de relative frekvensene skal bli?

Vedlegg 2:

13C Sammensatte forsøk

- Finne sannsynligheten i sammensatte forsøk med **programmering**.
- Sette opp de ulike **utfallene** i et sammensatt forsøk med tabell, liste og valgtre.
- Finne sannsynlighet i **sammensatte forsøk** ved å bruke uniform sannsynlighetsmodell eller multiplikasjon.
- Argumentere for om **spill** er rettferdige.

SPILL

HESTEVEDDELØP

Utstyr:

- To terninger
- Et spillebrett

Dette er et spill for 2, 3, 4, 6 eller 12 spillere.

Hver spiller velger etter tur bane på spillebrettet. Spillerne fortsetter å velge bane til alle 12 banene er valgt. Det kan bare være én spiller på hver bane.

Kast begge terningene. Legg sammen antall øyne på de to terningene. Den personen som har samme bane som summen av øynene på terningene, setter et kryss i sin bane. Spilleren som først kommer til mål med en av sine baner, vinner spillet.

MÅL	MÅL	MÅL	MÅL	MÅL	MÅL	MÅL	MÅL	MÅL	MÅL	MÅL	MÅL
1	2	3	4	5	6	7	8	9	10	11	12

Å SIMULERE SAMMENSATTE HENDELSER

Vi finner sannsynligheten for at summen blir minst 4 når vi kaster med to terninger.

Algoritme

Steg 1 Importerer biblioteket pylab.
Opprett variabelen **n** som står for antall ganger vi kaster terningene, og gi den verdien 100.
Opprett variabelen **gunstig** som står for antall ganger summen er minst 4, og gi den verdien 0.

Steg 2 Gjenta **n** ganger:
Kast **terning1** og **terning2** ved å trekke to tilfeldige tall.
Hvis **terning1** + **terning2** ≥ 4, skal **gunstig** øke med 1.

Steg 3 Skriv relativ frekvens på skjermen.

Den relative frekvensen er tilnærmet lik sannsynligheten når vi kaster terningene mange ganger.

Program i Python

```
1  from pylab import *
2
3  n = 100       # antall gjentakelser
4  gunstig = 0
5
6  for i in range(n):
7    terning1 = randint(1, 7)
8    terning2 = randint(1, 7)
9    if terning1 + terning2 >= 4:
10     gunstig = gunstig + 1
11
12 print("Relativ frekvens:", gunstig/n)
```

SNAKKE MATTE

Se på programmet ovenfor.

a Hvor mange ganger kastes de to terningene i eksemplet?

b Hva innebærer det at summen er minst 4?

c I hvilken linje skjer selve tellingen av de gunstige utfallene?

d Forklar hvilke programlinjer som hører til hvilke steg i algoritmen.

Vedlegg 3:

OPPDRAG

Forklare store talls lov med mange eksperimenter

Du har kanskje verken tid eller lyst til å kaste en mynt eller terning tusenvis av ganger? Datamaskinen er et godt verktøy når vi skal gjenta samme handling flere ganger. Ved hjelp av programmering skal vi se hvordan store talls lov fungerer. Utgangspunktet er situasjonen med myntkast som vi studerte på forrige side. Nå skal vi få en datamaskin til å gjenta myntkastet veldig mange ganger og finne gjennomsnittet.

Del 1 – Myntkastet

1 Vi må først definere at krone = 1, mynt = 0. Hva forventer du at gjennomsnitts-verdien blir hvis du kaster mynt og krone mange ganger?

2 Bruk programmet i Python til å gjennomføre 10 kast. Se hvordan gjennomsnittet utvikler seg.

3 Endre verdien av *N* til 50, 100 og 1000. Beskriv hvordan kurven utvikler seg.

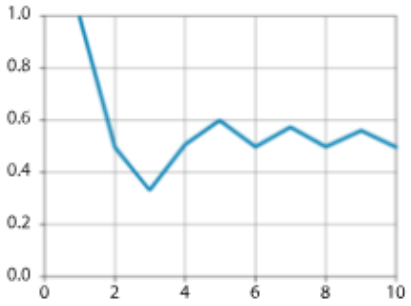
Del 2 – Summen av to terningkast

1 Lag en oversiktstabell som viser alle mulige utfall av summen av to terningkast. Bruk tabellen til å sette opp en hypotese over hvilken sum dere tror vil være et gjennomsnittlig resultat for mange kast med to terninger.

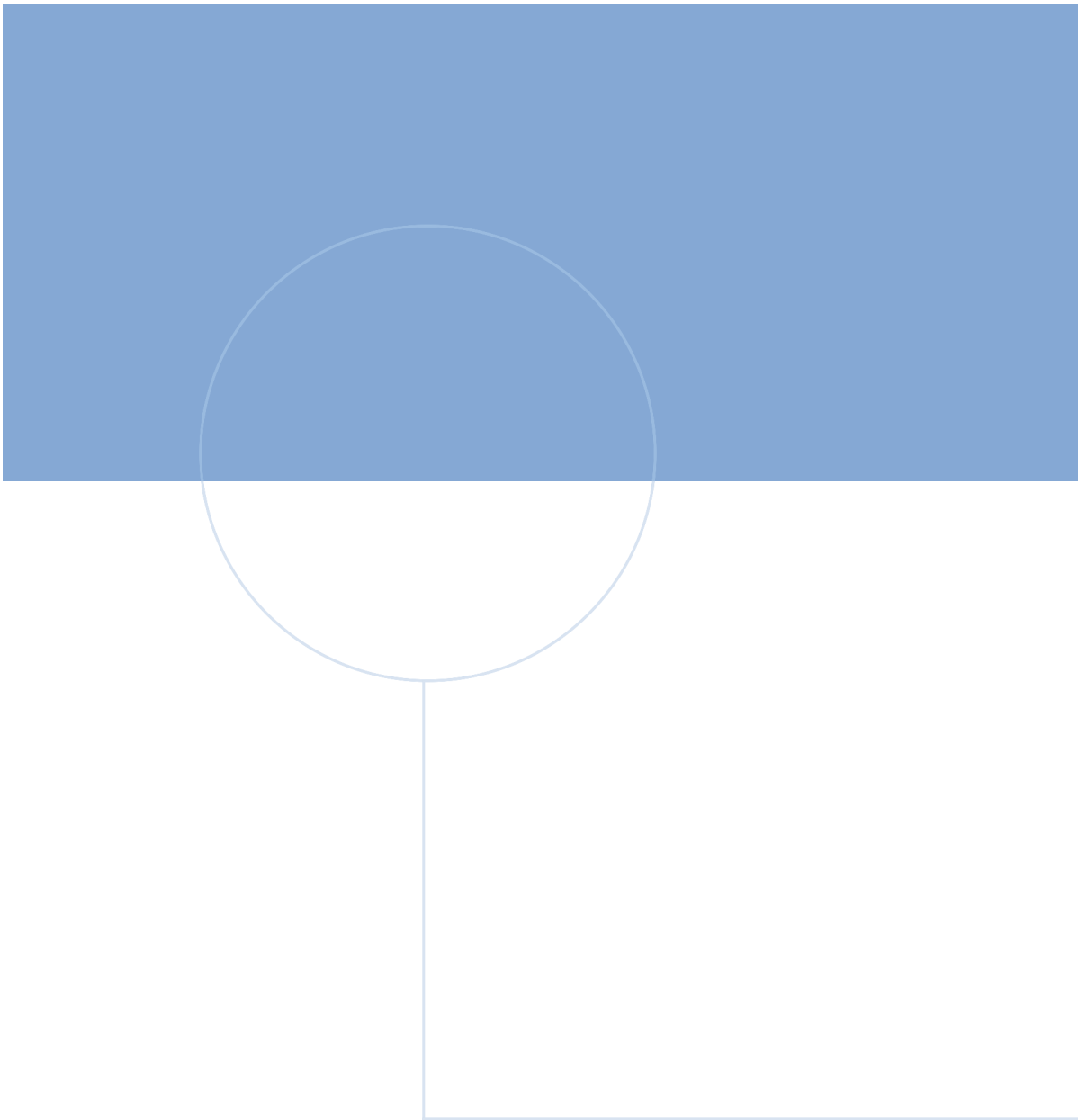
2 Gjør endringer i programmet slik at det kan gjennomføre to terningkast med verdier 1–6 og summere disse. Lag plot for 100, 1000 og 10 000 doble terningkast.

3 Stemmer hypotesen deres?

```
from pylab import *

# N = antall myntkast
N=10

# Lager en liste med N myntkast,
# hvert kast får verdien 0 eller 1
kast=randint(0,2,N)

# lager en liste med kastnummer (x-verdier)
x=arange (1,N+1)

# lager liste med kumulativt
# gjennomsnitt (y-verdier)
gjsnitt_kast=cumsum(kast)/x

plot(x,gjsnitt_kast)
axis([0,N,0,1])
grid()
show()

```



Eksempel på resultat etter 10 myntkast.



NTNU

Kunnskap for en bedre verden