Lesbar programmering inn i bioteknologi-utdanningen

Essay i PPUN400 Universitetspedagogikk for vitenskapelig ansatte

Jon Olav Vik 2018-12-07

jon.vik@nmbu.no • nmbu.no/emp/jon.vik

1. Bakgrunn: STIN100 Biologisk data-analyse

Grunntanke: Fortrolig med utforsking og etterrettelig analyse av data

Lesbar programmering er Dag Langmyhr sin gjendiktning av Knuths (1984) begrep *literate programming*: Sømløs blanding av dataprogramkode og fri, forklarende tekst, som kan veves sammen med output av programmet til en selvdokumenterende, etterprøvbar rapport.

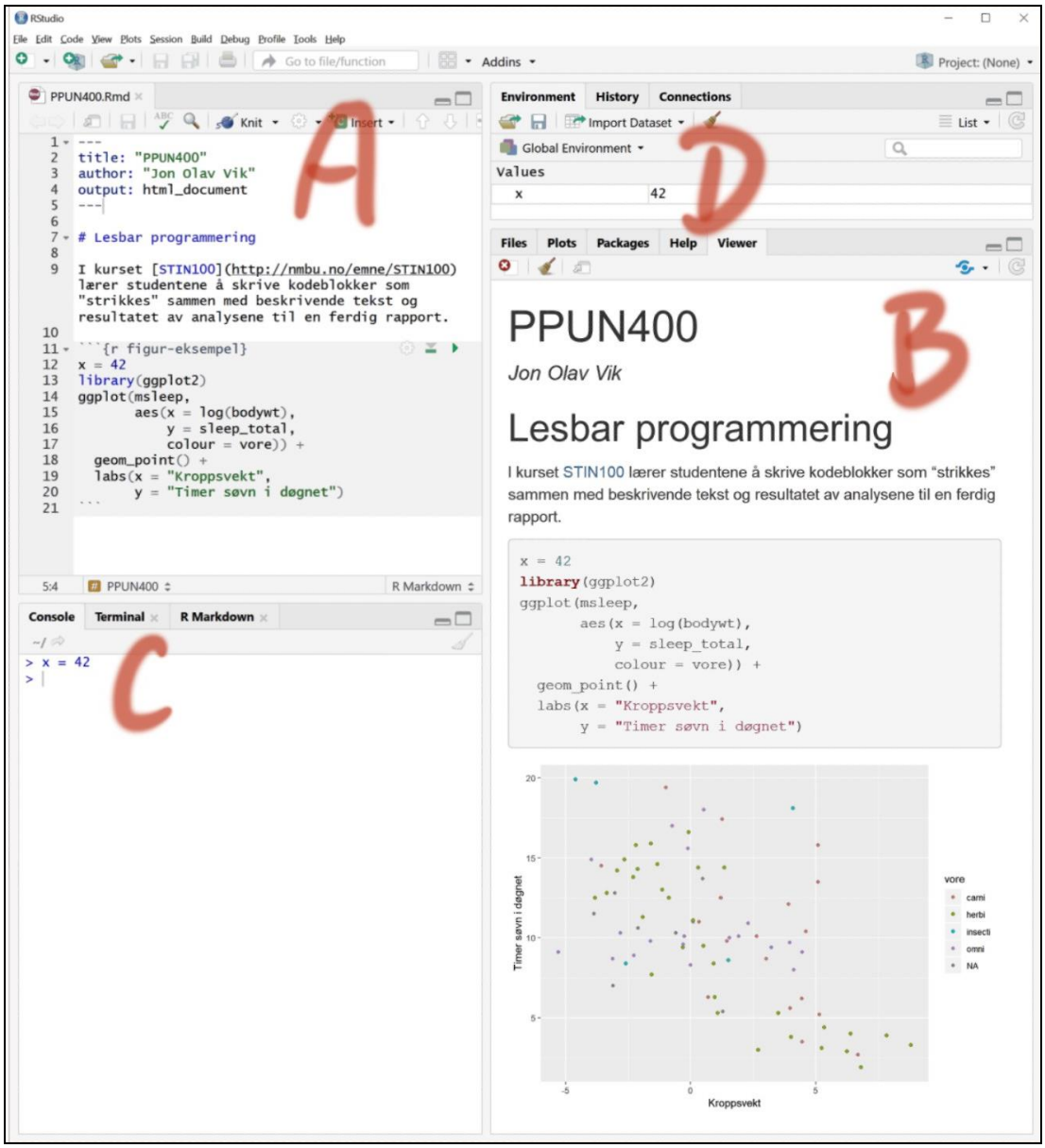
Kurset STIN100 er nytt høsten 2018 og er motivert av et ønske om at masterstudenter flest skal bli mer selvgående til å håndtere og analysere data. Kongstanken er å ta tidlig tak i studentene og gjøre dem komfortable med å se på, prosessere og beskrive data og hva dataene betyr biologisk. Til dette bruker de programmeringsspråket R og lesbar programmering gjennom RStudio (Figur 1) og R-pakken knitr[1] (Xie 2015) -- de samme verktøyene som vi foreleserne bruker i vårt eget vitenskapelige arbeid. Kunnskap for livet, altså, helt i tråd med NMBUs læringsfilosofi (NMBUs studienemnd 2015).

Slik automatisert rapport-generering er gull verdt for etterrettelig forskning (Gandrud 2015). Man trenger ikke lenger frykte at man har glemt å lime inn siste versjon av resultatene: Dette gjøres automatisk hver gang rapporten regenereres. Er man skeptisk til antagelsene, kan man kopiere kildedokumentet, endre forutsetningene, og umiddelbart se hvordan resultatene påvirkes. Denne arbeidsmåten, sammen med revisjonshåndtering, er etter mitt syn et svært viktig tiltak mot *reproduserbarhetskrisen* i vitenskapen (Peng 2011) og for framtidige arbeidsgiveres tillit til studentenes analyser.

Reproduserbare analyser har en rekke positive didaktiske bivirkninger som kanskje ikke er åpenbare for folk som ikke analyserer data til hverdags. Det er raskere og lettere å veilede når du kan *se* analysen kjøre og kanskje bære galt avsted. Man kan reprodusere problemet, utforske det interaktivt, og dele løsninger og anbefalinger ved å sende et revidert kildedokument tilbake. Samtidig blir studentene trent til å lage konkrete, fyllestgjørende, reproduserbare problembeskrivelser, noe om forventes på verdifulle spørsmål-og-svar-nettsteder som Stack Overflow.

---

[1] Nerd alert: Navnet *knitr* er et ordspill på *neater og knitter*, som et alternativ til *weaving*, som er Knuths opprinnelige navn på sammenflettingen av tekst, kode og resultater. Ordspillet fungerer nok best med kinesisk aksent.



Figur 1. Lesbar programmering i RStudio. Brukeren skriver en blanding av fritekst og programkode (A), som automatisk flettes sammen ("strikkes") til en rapport (B). Samtidig går en aktiv interaktiv R-sesjon i konsollen (C), og man har oversikt over definerte variabler til enhver tid (D). Andre faner kan vise plott, filsystem, hjelpetekster og andre funksjoner.

Rammebetingelser for kurset

Kurset er 10 studiepoeng og går over 14 uker med 2 timer forelesning og 2 x 2 timer øvinger. Det er tre obligatoriske gruppe-innleveringer og en avsluttende eksamensoppgave (bestått/ikke bestått). Vi er tre forelesere: Kurset ble opprinnelig unnfanget av Torgeir Hvidsten (kursansvarlig) og Simen Sandve, mens jeg ble med fra februar 2018 av. Sammen er vi et sammenknyttet og entusiastisk team. Tre masterstudenter er øvingslærere.

STIN100 er et lavterskelkurs og krever ingen forkunnskaper. Foreløpig er det obligatorisk kun for førsteklassinger som tok den mest avanserte matematikken på videregående, siden disse

slipper å ta MATH100 og har en garantert luke i timeplanen. Men på sikt er STIN100 tenkt å bli obligatorisk for alle studenter på bioteknologi, slik at andre kurs kan forutsette at studentene vet å orientere seg i data, bearbeide dem for å besvare biologiske spørsmål, og redegjøre for analysen på en etterrettelig måte.

Vi forventet å få stort sett studenter som hadde STIN100 obligatorisk, og trodde dette var ca 15 stykker. Det viste seg å bli 62 oppmeldte, de fleste fra andre og tredje studieår. Oppunder femti fullførte eksamensprosjektet, noe vi er svært fornøyd med.

2. Planlegging: Grunnferdigheter, dybdeøvelser, sluttprosjekt

Grunnprinsipper

Vi valgte å fokusere primært på hva studentene skulle kunne *gjøre* på egenhånd etter avsluttet kurs. Aktiv og praksisbasert læring (Pettersen 2005, 121) er altså bærende elementer i opplegget. Videre vil vi at studentene skal tilegne seg strategier for å lære mer etter behov: Data-analytikere driver kontinuerlig etterutdanning med verktøy som integrert programdokumentasjon, Google-søk og spørsmål-og-svar-nettstedet Stack Overflow. Kurset er lite fasit-orientert, siden det oftest fins mer enn én relevant måte å presentere og analysere data på. Dette medfører *deltakerstyring* (Pettersen 2005, 186) gjennom at noen studenter kan gå mer i dybden på biologi og tolking av data, mens andre fokuserer på det presentasjonsmessige eller programtekniske.

Samtidig skal studentene lære viktige *begrepsapparater* for kurstemaene, både for å strukturere tankeprosessen, og for å kunne analysere andres plott og analyser og sette ord på hva som gjør dem gode eller mindre gode. To eksempler på sentrale begreper i datavisualisering er *mapping*: avbildningen av variabler fra en dataramme på fysiske størrelser som x-posisjon, punktstørrelse eller farge; og *geom*: bildeelementet som representerer rå eller aggregerte data, f.eks. punkter, linjer, histogrammer. To sentrale begreper i multivariat analyse er *avstand* (mellom to sett målinger av mange variabler) og *projeksjon* (å vise "skyggen" av høydimensjonale data i færre dimensjoner). Disse er illustrert i Figur 2.

Begrepene innarbeides helst gjennom *aktiv bruk*. Etter at de er introdusert, brukes de igjen og igjen i spørsmål og oppgaveformuleringer, og etter hvert lærer studentene å bruke begrepene når de stiller spørsmål, beskriver hva de har gjort, eller hva resultatene viser. Aktiv bruk av språket er også sentralt i studentenes mentale begrepsdannelse (Vygotskij 1934). Konstruktiv innretting av læringsaktivitetene til de ønskede sluttferdighetene (Biggs 2011) er også vist å gjøre læring raskere og mer overførbar til praktisk anvendelse.

På denne måten spenner vi hele den kognitive skalaen i Blooms (1956) taksonomi, fra terminologikunnskap og forståelse til aktiv bruk, analyse, syntese og evaluering. Jeg anser disse for å være komplementære mer enn et sett stigende nivåer. Blooms "følelsesdomene"

er mer av en stige, og handler om verdispørsmål. Jeg føler ikke dét er så relevant for våre studenters læring, annet enn at jeg håper at de etter fullført kurs føler en instinktiv avsky for ikke-reproduserbare analyser, manglende akse-forklaring og uhensiktsmessig bruk av grafiske virkemidler. Det psykomotoriske domenet er derimot viktig, idet dataanalyse og programmering er veldig praktiske ferdigheter på sitt vis. Man må aktivt *interagere* med dataene, samarbeidspartnere, datamaskinen og internett for å "få inn i fingrene" hvordan man legger og gjennomfører strategier for analyse, utvikling og feilsøking.

Programmerings-didaktisk støttet vi oss mye på "Ten quick tips for teaching programming" (Brown and Wilson 2018). Videre vektla vi å trene opp *strategier* og ikke bare *kunnskap* (Robins, Rountree, and Rountree 2003). Det er også svært vanskelig å egentlig skjønne hvordan et program *virker* uten å kunne *skrive* det selv. Ved å velge ut et lite antall funksjoner som spenner over mange biologiske spørsmål, håpte vi at STIN100-studentene skulle kunne få dyp kunnskap med overføringsverdi til deres senere egenlæring.

En avgrensing vi gjorde, var å forutsette at alle datasett forelå på lett tilgjengelige formater og var fri for tekniske feil. I virkeligheten brukes vanligvis en latterlig stor andel tid på å renske data før man i det hele tatt kommer i gang med analysen. Men å gjøre den renskingen krever ofte lang erfaring med både programmering, feilsøking og hva variablene betyr, og er uegnet som nybegynneroppgave.

Kjøreplan for kurset

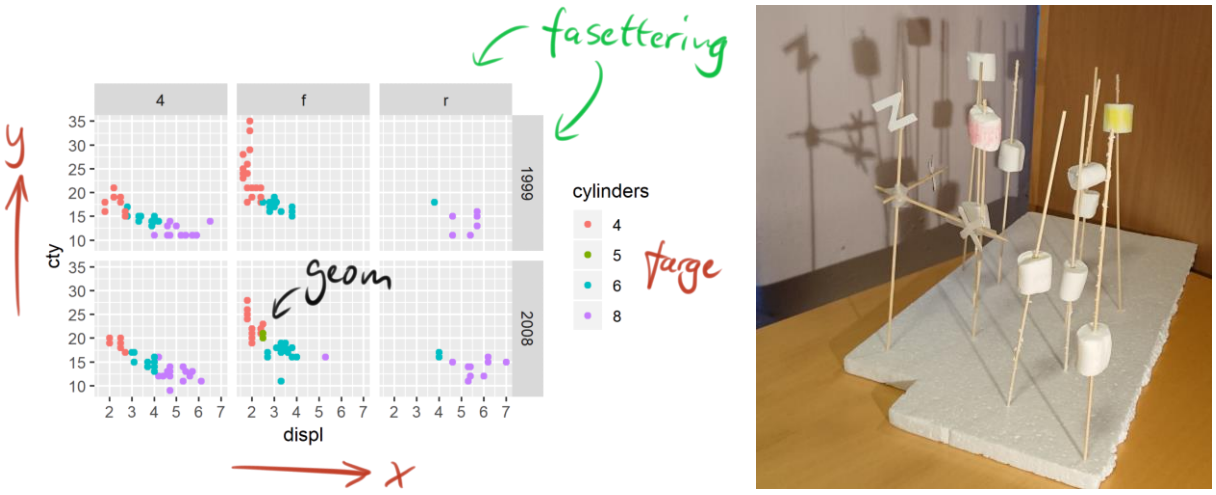
Første forelesning ga en kort oversikt over kursopplegget (Tabell 1) og lærte bort grunnferdigheter i bruk av programmeringsverktøyet, samt de to viktigste kodebibliotekene og tilhørende begrepsapparat: ett for datavisualisering (Figur 2) og ett for å bearbeide data.

Deretter fulgte tre dobbelt-uker med case-fokuserte (Pettersen 2005, 169) dypdykk i forskningsdata fra NMBU. Eksterne forskere stilte opp og fortalte om sine biologiske spørsmål, hvordan de utformet studiene sine, og hvordan dataene var generert. Forskerne hadde også gitt oss eksempler på framstilling og analyse av dataene, som vi bearbeidet til programmerings-øvinger. Studentene jobbet med disse i par. Fordypningen ga studentene intuisjon om dataene, samt tid til å innarbeide et aktivt repertoar av teknikker for visualisering, oppsummeringer og dataanalyse.

Øvingsdokumentene startet med et illustrert motivasjonsavsnitt og en boks med læringsmål. Øvingene var selv generert med lesbar programmering (Figur 3), noe vi håpte motiverte studentene ved å illustrere de rike mulighetene i verktøyet de var i ferd med å lære. Kodeblokkene var skjult bak en "vis kode"-knapp, slik at studentene kunne prøve selv, men se løsningsforslag hvis de satt fast. Til senere øvinger slo vi av opsjonen for å inkludere noen av kodeblokkene. En del studenter syntes dette hjalp dem å prøve ordentligere på egen hånd.

Tabell 1. Kursplan for STIN100 Biologisk data-analyse.

Uke	Tema	Ansvarlig
36	**Introduksjon til kurset** Hvor vi introduserer dere til multidimensjonelle data og programmering i R/RStudio. Vi gir dere eksempler på hva dere kommer til å lære i kurset.	Torgeir
37	«**Datagrafikkens grammatikk**» Hvor vi lærer å lage bilder av data så vi kan forstå dem bedre. Vi lærer hvilke elementer som datafigurer er bygd opp av, og regler for hvordan de kan kombineres. library(ggplot2)	Jon Olav
38	**Datatyper og strukturer** Hvor vi lærer om ulike data og hvordan disse kan lagres i ulike strukturer. Vi ser også på statistiske sammendrag, hypotesetester og hvordan vi kan simulere data.	Torgeir
39	**«Split, apply, combine»: Oppsummere delmengder av datasett.** Hvor vi lærer mer om statistiske sammendrag. library(dplyr)	Jon Olav
40	**Datasett 1: GPS-data**	Jon Olav
41		
42	**Datasett 2: Genregulering**	Simen
43		
44	**Datasett 3: Tarmflora**	Torgeir
45		
46	**Prosjekt**	
47	Fordypning i ett av datasettene.	
48		
49	**Eksamen: Innlevering av skriftlig rapport («artikkel»)**	



Figur 2. Viktige begrepsapparater i STIN100 Biologisk data-analyse: Datagrafikkens grammatikk (venstre, etter Wickham 2009) og dimensjonsreduksjon av multivariate data (høyre), her illustrert ved marshmallows projisert på vegg ned fra dimensjonene X, Y og Z.

3.1 En nærmere titt på `foxes.sp`

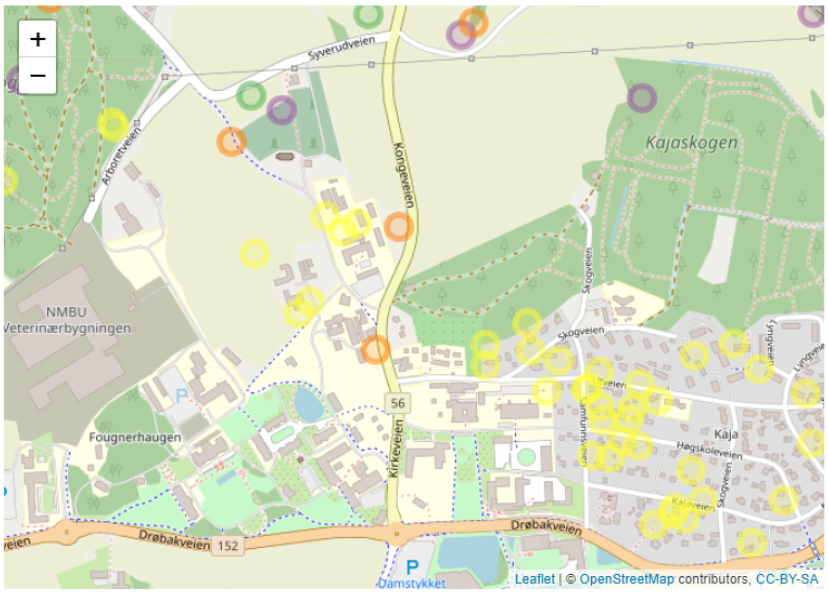
GPS-dataene for de sju revene ligger i `foxes.sp`.

Skriv inn følgende kode og kjør den, så skal du lære hvordan den virker senere i øvingen:

Hide

```
# Lag en funksjon som oversetter reve-navn til farge.
pal <- colorFactor(palette = "Set1", domain = foxes.sp$id)

foxes.sp %>%                                    # Ta reveposisjonene, og:
  spTransform("+proj=longlat +ellps=WGS84") %>% # Oversett til lengde/breddegrad
  leaflet %>%                                   # Lag interaktivt kart
  addTiles %>%                                  # Legg på grunnslagskart
  addCircleMarkers(color = ~pal(id))            # Legg på posisjonsmerker farget etter rev
```



Klikk og dra kartet, og klikk `+` og `-` for å zoome inn og ut. Finn NMBU på kartet, og gjerne huset ditt. Vilt kult, ikke sant?

Pakka `leaflet` er et enkelt og kraftig kartverktøy som er gratis og virker både i RStudio og ferdigknittet HTML.

Datasett_1_romlige_data.Rmd | foxes

Filter

	date	id	geometry
2189	2018-08-10 02:33:25	fox_12	c(10.7639120442696, 59.6779199701555)
2209	2018-08-10 02:49:47	fox_12	c(10.7668550442571, 59.6773449701634)
2228	2018-07-27 19:43:28	fox_13	c(10.7652800442655, 59.6786949701608)
2234	2018-07-27 19:45:56	fox_13	c(10.7652420442657, 59.6786779701607)
2235	2018-07-27 19:46:12	fox_13	c(10.7652420442657, 59.6786779701607)

Figur 3. Øvingsdokument (øverst) og datasett (nederst) om GPS-merket rev i NMBU-området. Kartet er interaktivt og kan flyttes og zoomes. Studentene får det samme på sin egen maskin ved å kjøre den viste koden (som kan skjules igjen med knappen "Hide").

De siste fire ukene utgjorde eksamen, som ble gjort i samme grupper og vurdert med bestått/ikke bestått. Her skulle studentene skrive en rapport på noe nær vitenskapelig artikkel-form, og ha med minst én ny analyse eller figur. Her var det altså viet mer tid til biologisk tolking og formidling.

Parprogrammering og dens didaktiske fortrinn

Parprogrammering er samarbeid mellom to (til nød tre) studenter. Den ene ("sjåføren") har tastaturet og skriver programkode, mens den andre ("navigatøren") observerer, kvalitetssikrer og kommer med forslag. Rollene byttes ofte. Denne formen for aktiv læring er vist å bedre innlæring, selvtillit og kodekvalitet (McDowell et al. 2006) og er ofte triveligere enn å kode alene (Williams et al. 2000). Dette gjelder særlig overfor problemer/anvendelser som er nye for programmererne, og gevinsten ved paring er størst der én eller begge er nybegynnere (Lui and Chan 2006).

Dette er nettopp tilfelle i STIN100: Problemstillingene er nye for studentene enten biologisk eller programmeringsmessig sett, men noen studenter har mer data-erfaring enn andre. Diskusjon om framgangsmåter hjelper til å innarbeide begreper ved å formulere dem muntlig og hente dem fram i relevante situasjoner, f.eks. hvilke data som er relevante for et gitt biologisk spørsmål, hvilken programfunksjon som kan implementere et skissert plott, eller hvordan data må transformeres for å muliggjøre en bestemt sammenligning.

Parprogrammering er en form for *samarbeidslæring* (Johnson, Johnson, and Smith 2014) der man "spiller hverandre gode" snarere enn å konkurrere. Den sosiale relasjonen og gjensidige avhengigheten mellom makkerne bidrar til å holde fokus, og den språklige interaksjonen støtter begrepsinnlæring som nevnt over. Felles utfordringer viser også at man ikke er alene om å gjøre feil eller føle seg overveldet, noe som støtter opp om troen på at man kan mestre læringsmålene. Samarbeidstonen er viktig for at dette skal fungere godt, og vi var aktivt på vakt og ute blant pultradene for å se an hvordan dette gikk i praksis.

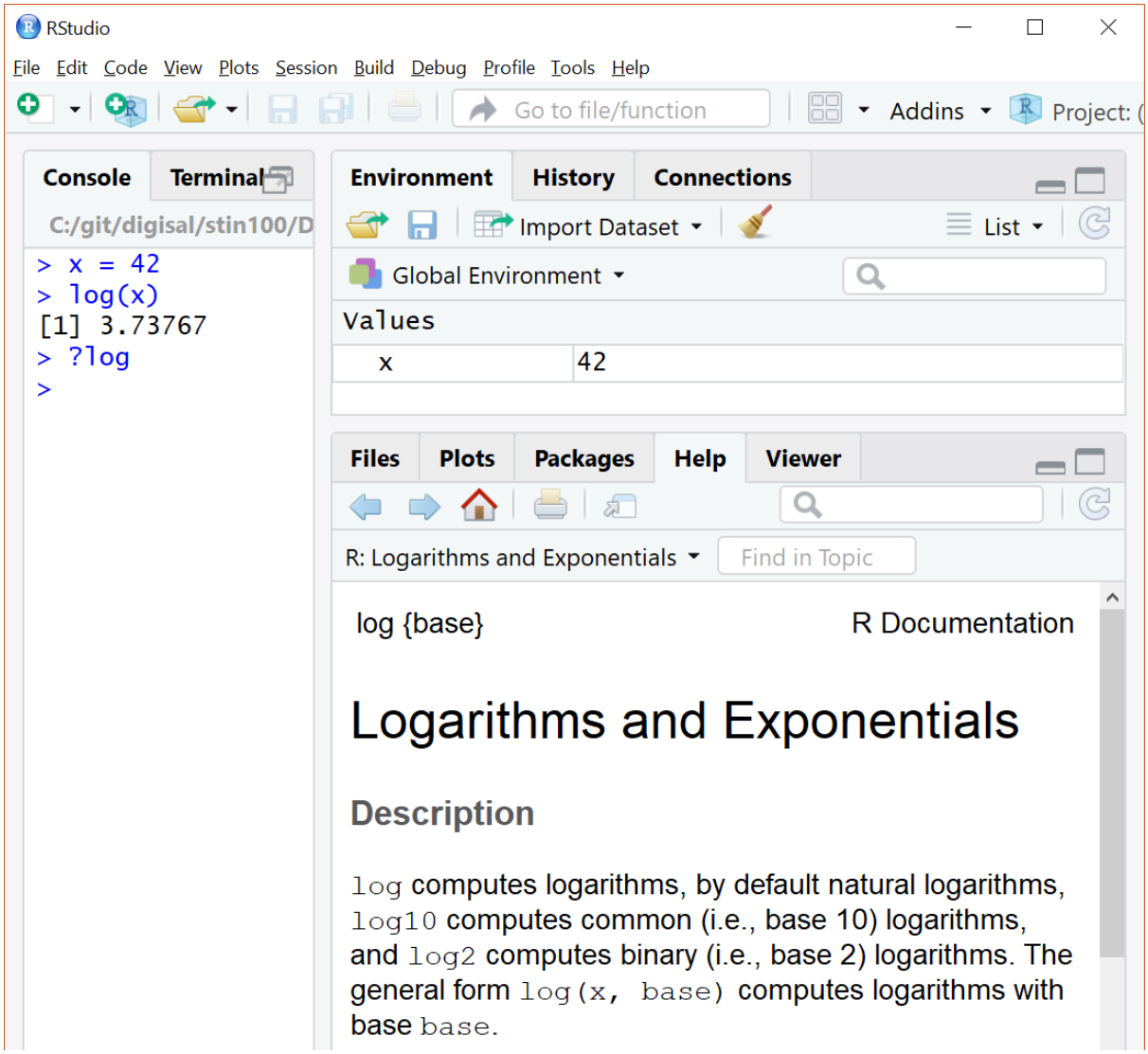
3. Praktisk gjennomføring: Live koding, øvinger, veiledning

I første forelesning presenterte vi oss selv og forklarte at STIN100 lærer bort samme arbeidsmåter og verktøy som vi selv selv bruker på ordentlig. Vi snakket om hvilken utdanning og nåværende jobber vi har, og håpte dermed å skape engasjement og relasjon mellom studentene og oss. Det var en fordel at vi har litt ulike bakgrunner og legninger, så vi kunne karikere hverandres styrker og svakheter og dermed forsikre studentene om at man kan få til mye selv om ikke alt er perfekt.

De innledende ukene gikk fra det kjente til det ukjente: Bruke datamaskinen som kalkulator, variabler for å ta vare på resultater og bruke dem videre, flere datatyper inkludert tall, tekst, sant/usant, vektorer (mange tall av samme sort) og datarammer (en tabell der hver kolonne har et navn, og alle elementene i kolonna har samme datatype).

Datarammer er sentrale fordi de favner mange praktiske brukerkasus, og er den forventede datastrukturen for de fleste R-kodebiblioteker. STIN100 vektlegger library(ggplot) for plotting og library(dplyr) for utvalg/oppsummering/transformasjon av data. Disse er to av pakkene i et større rammeverk som kalles tidyverse (Wickham and Grolemund 2016), som studentene dermed lett kan tilegne seg resten av.

Studentenes første møte med R-programmering var live koding (Rubin 2013), hver student på sin laptop som gjorde etter det læreren viste. De kunne se hvordan "x = 42" resulterte i at variabel-oversikten fikk en ny oppføring "x" med verdien 42, som viste seg å være et heltall. Da de kom til funksjoner, kunne de slå opp hjelp for funksjonen direkte inni RStudio. Ved å gjøre dette selv, ble de kjent med hvor i RStudio de ulike verktøyene finnes og hvordan de hentes fram.



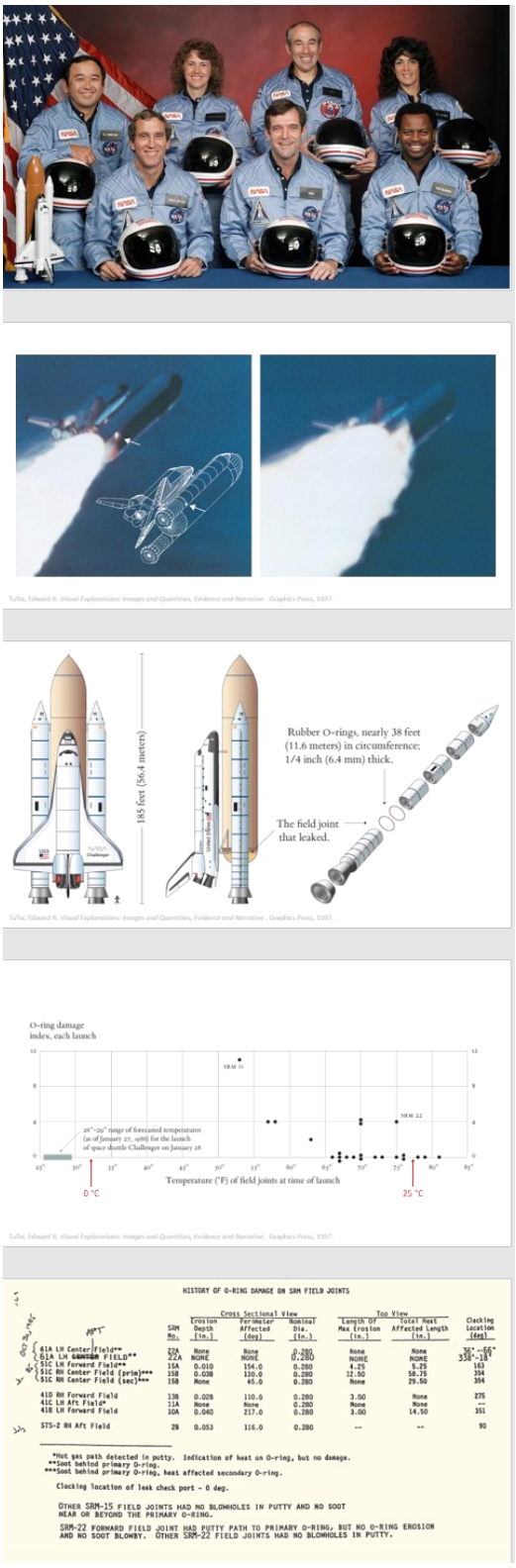
Figur 4. Et enkelt eksempel på live koding. Hver student gjør etter læreren på egen laptop.

Forelesningen om datagrafikkens grammatikk startet med et rystende eksempel fra virkeligheten (Figur 5): Da romferja Challenger eksploderte på direktesendt TV (CNN 1986). Saken er et berømt eksempel på viktigheten av å kommunisere data slik at de blir lette å resonnere om (Tufte 1997). Feilen som sprengte romferja var varslet på forhånd, men alvoret kom ikke fram i tabellene og presentasjonene som ble brukt. STIN100-studentene skal bli brennende engasjert i å kommunisere data effektivt, sannferdig og lettforståelig.

Det var mange anledninger der skeptiske studenter lyste opp av mestringsfølelse, særlig når en grafisk framstilling eller rapportgenerering endelig virket, og de kunne *se* store informasjonsmengder i egenlagde framstillinger. Flere fant ut at de hadde hatt revebesøk i hybelhagen (Figur 3), og én dro til og med ut i skogen med GPS for å lete etter spor av revehi. (Han fant det ikke, dessverre.)

Øvingsmaterialet måtte vi i stor grad lage selv, siden vi blander fag som tradisjonelt undervises separat. Tidyverse-pakkene har utmerkede læremidler på nett, men eksemplene deres er knusktørre; flyavgangstider og sånt. Vi fikk gode tilbakemeldinger på at dataene kom fra virkeligheten og fra studentenes eget universitet. Formodentlig bidro dette til engasjement og konstruktiv innretting i deres læring.

I praksis ble øvingene forberedt litt i siste liten, noe som var et (fullt forståelig) stressmoment for øvingslærerne. De skulle gjerne hatt bedre tid til å gjenoppfriske relevant kunnskap og dels lære nye ting. Dette blir bedre til neste år.



Figur 5. Lysbilder fra forelesningen om datagrafikk (Tufte 1997, lisensiert for undervisning). Øverste bilde: https://en.wikipedia.org/wiki/File:Challenger_flight_51-l_crew.jpg

Gruppeinndeling til øvingene fulgte først-til-mølla-prinsippet, så de fleste var i par med kjentfolk de hadde valgt selv. Etternølerne fordelte vi alfabetisk, og en måtte omfordeles fordi makkeren trakk seg fra kurset.

Stemningen på parøvingene var god fra første stund: Lavmælt, fokusert diskusjon om både dataverktøyet og oppgavene. De kursansvarlige var mye til stede for å se hvordan det hele fungerte, noe som kom godt med fordi tre øvingslærere var litt lite i starten.

Studentene ble bevisst kastet ut på dypt vann ved at øvingen krevde at de søkte seg fram til ting som ikke var gjennomgått på forelesning. Et par av øvingslærerne var svært skeptiske, men studentene tok det etter mitt skjønn på strak arm. Å lære å lære er en nøkkelferdighet, og en vei man må gå *selv*. Ikke dermed sagt at man må gå *alene*: Når studentene ba om hjelp til å finne ut av ting, satte vi oss ned sammen med dem, fikk dem til å sette ord på hva problemet var, beskrev med våre egne ord hvordan vi gikk fram for å avgrense, identifisere og gjenskape problemet, og hvordan vi lette etter veier videre. Dette innebar f.eks. å finne søkeord til google, lete i programdokumentasjon, eller grave seg nedover i datastrukturene til det kjørende programmet. Dette ble altså en form for mesterlæring/modellering (Pettersen 2005, 85) i *problemløsingsstrategier*, mer enn å dele ut svar.

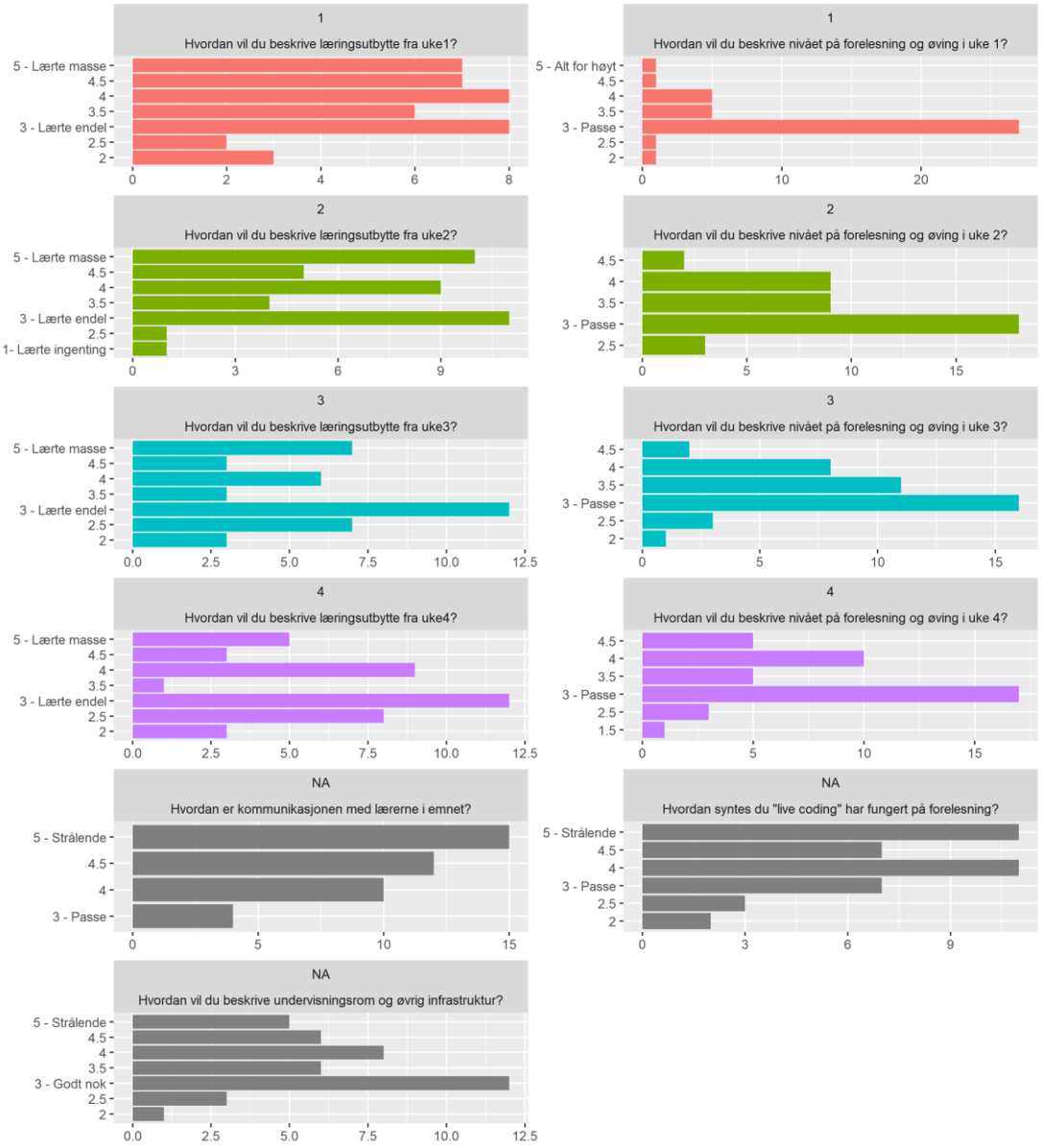
Parprogrammeringen fungerte ikke helt som tiltenkt, men likevel bra. De fleste ville gjerne ha en versjon av Rmd-dokumentet på sin egen laptop, slik at begge gjerne satt og skriv inn samme kode. Mange av de positive effektene var der fortsatt: Det var lettere å innse løsninger på problemer når man formulerer det i ord (Vygotskij 1934), flere idiotfeil ble unngått, og arbeidet skapte samhold i gruppa.

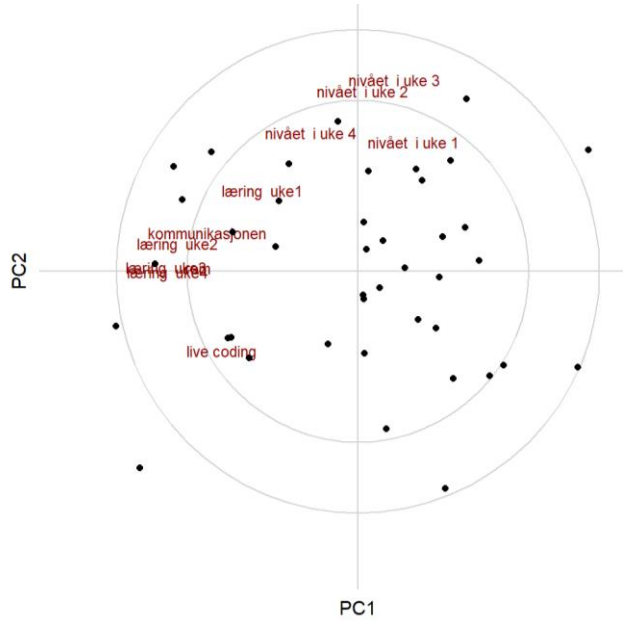
4. Vurdering av tiltaket: Egen- og studentevaluering

Studentenes evaluering av kurset var svært positiv når det gjaldt læringsutbytte og kontakt med foreleserne (Figur 6). En del av øvingene var det noe frustrasjon over (Figur 7). Mitt inntrykk er at dette særlig gjaldt forelesninger/øvinger der vi hadde knapt med tid til forberedelse; de mest gjennomarbeidede oppleggene unngikk mange frustrasjonsfeller.

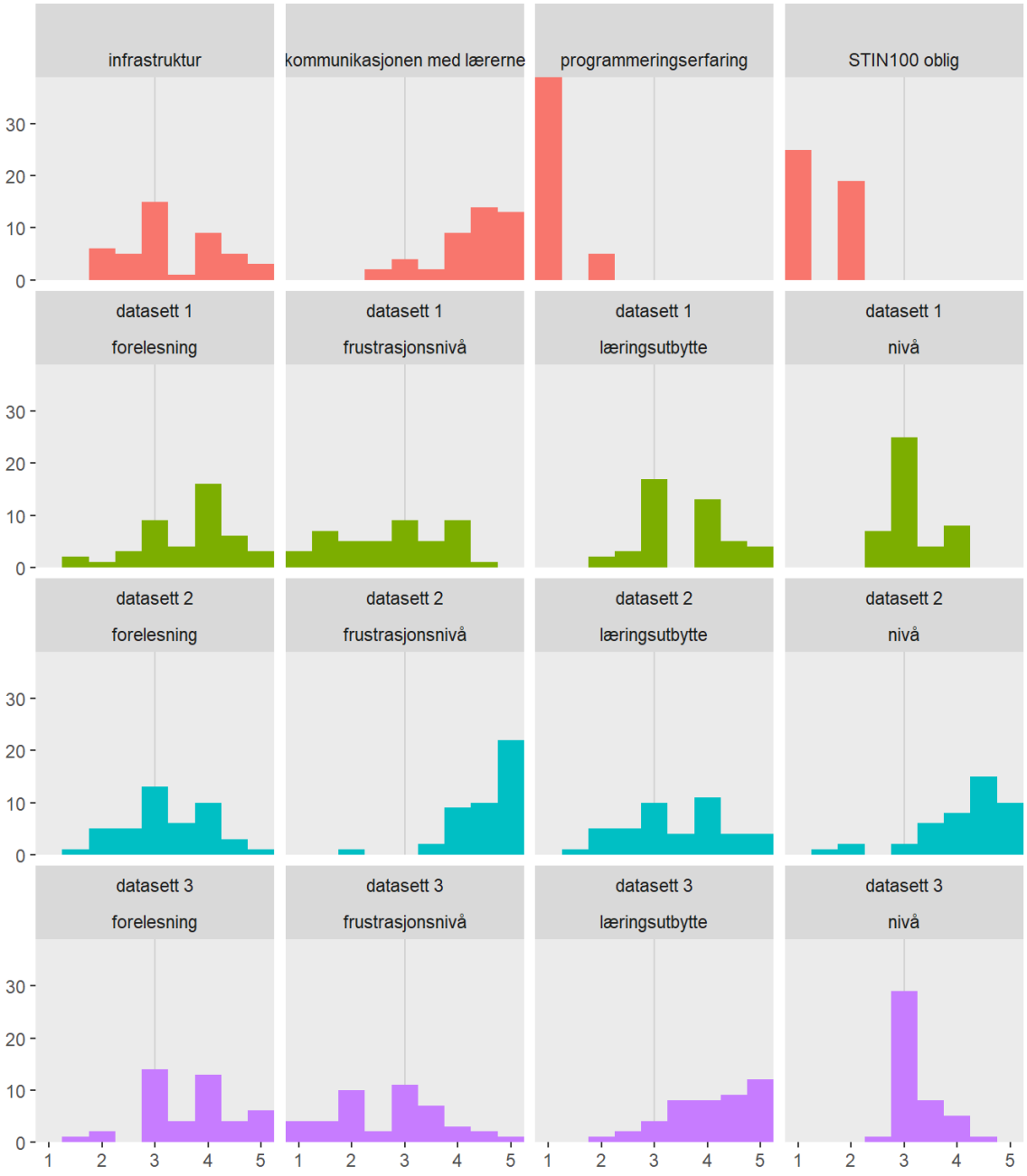
Tekstkommentarene (ikke vist) var jevnt overstrømmende positive; det virket som studentene satt pris på det vi tilbød og forsøkte å få til, og tilga oss de feilskjærene som nødvendigvis måtte komme innimellom.

De gode evalueringene var særlig gledelig gitt studentenes mange ulike bakgrunner, ambisjonsnivå og mål. Jeg tar det som et tegn på at vi lykkes i å tilby mange utviklingsretninger -- mer biologi for dem som vil dét, mer programmering eller visualisering for andre.





Figur 6. Underveisevaluering av introduksjons-modulen. Øverst: Karakterfordeling. Nederst: Samvariasjon mellom karakterer på delspørsmål, projisert ned i to dimensjoner. Hver prikk er en student og hver tekst er et delspørsmål. Horisontalaksen korrelerer sterkt med opplevd læringsutbytte (høyere = venstre), mens vertikalaksen korrelerer med opplevd vanskelighetsnivå. Studenter langt til høyre opplevde altså lavere læringsutbytte enn gjennomsnittet, men gjennomsnittet var altså godt over "lærte endel".

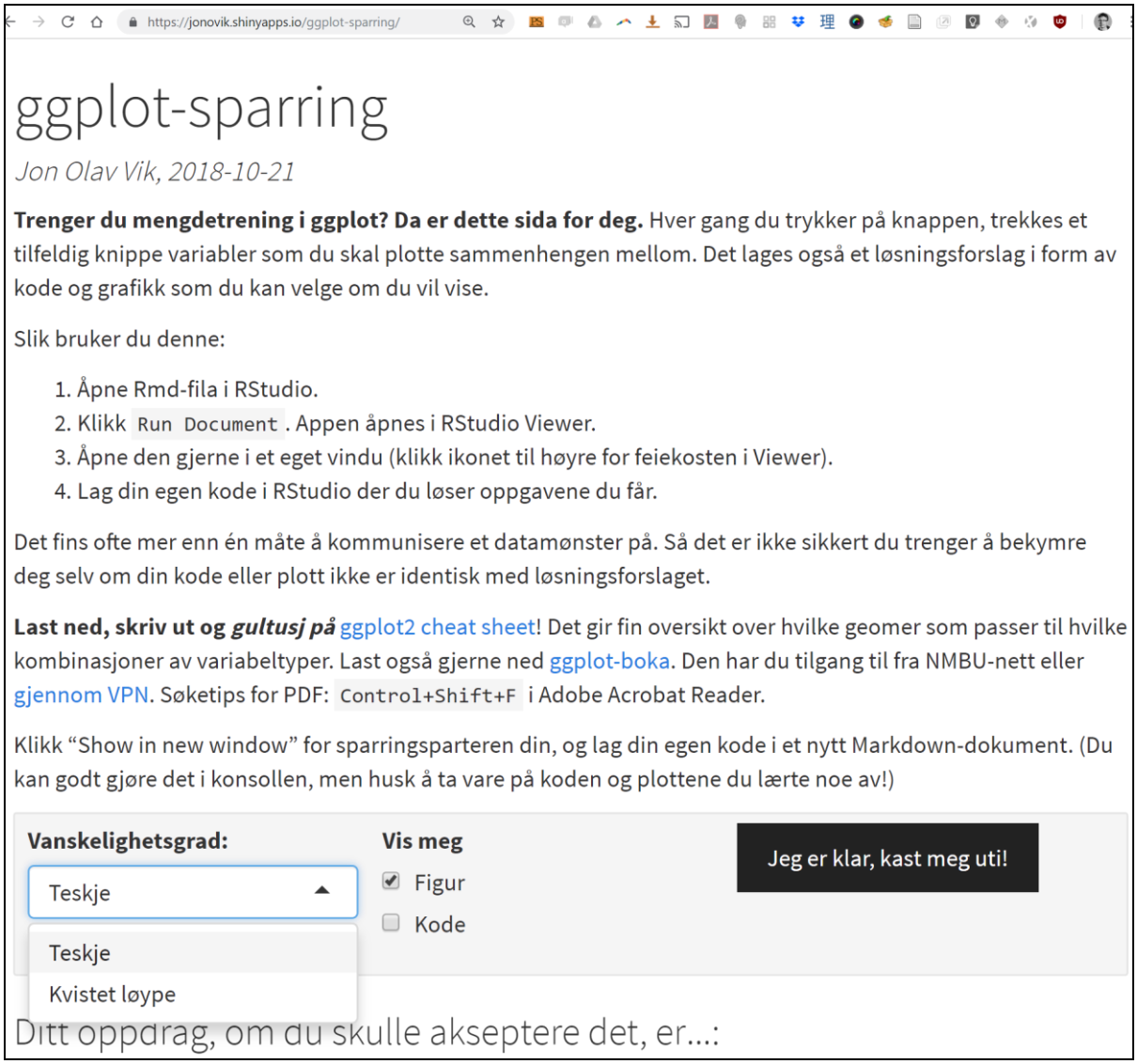


Figur 7. Underveisevaluering av modul 2, datasett-øvingene. Streken angir karakter 3, "passe". Hvorvidt studentene hadde tidligere programmeringserfaring, og hvorvidt kurset var obligatorisk for dem, er kodet som 1 = nei og 2 = ja.

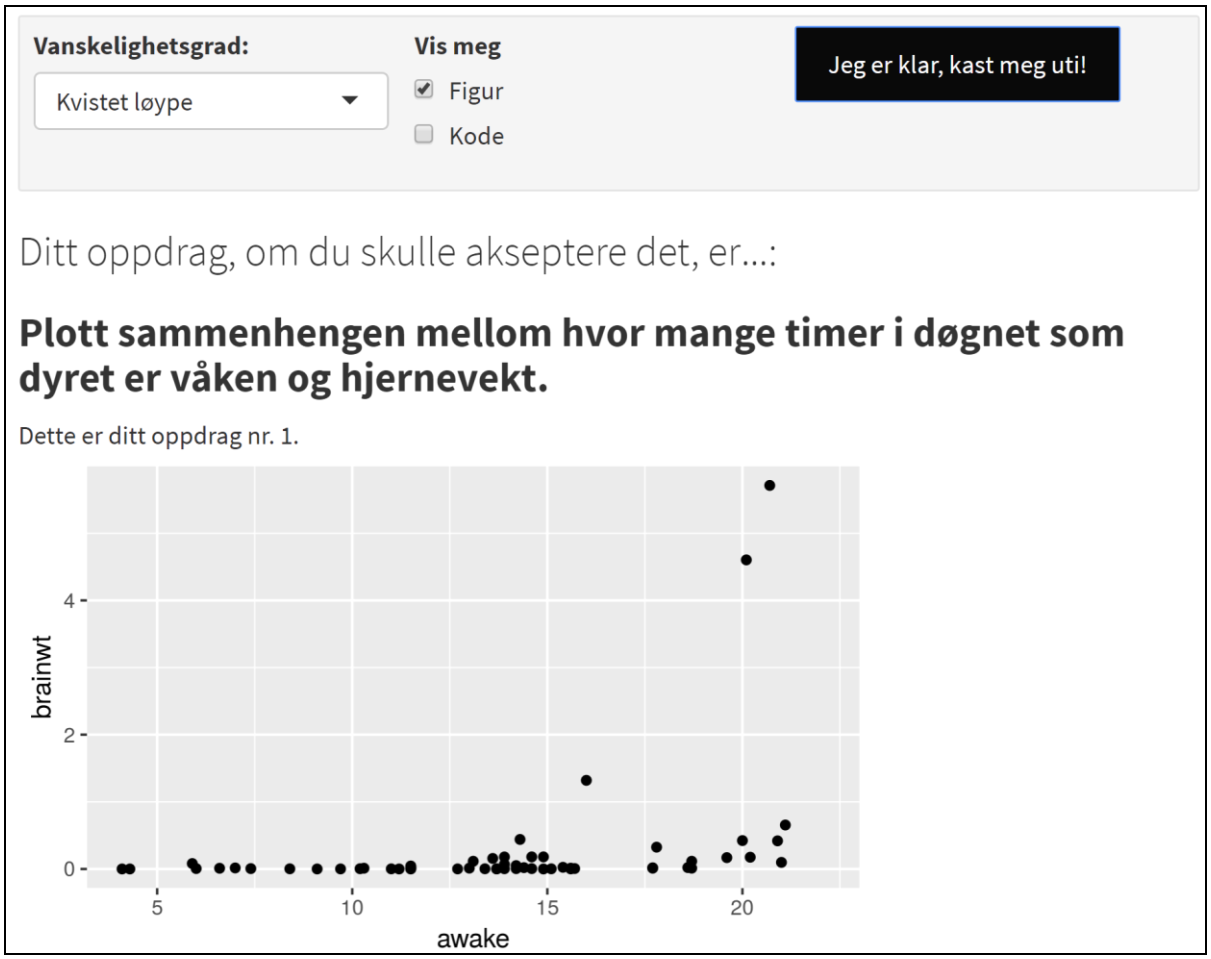
Entusiasmen var større på læringsaktiviteter vi hadde laget selv enn på rutineøvinger fra R-dokumentasjonen, skjønt begge var nyttige og nødvendige. Det er imidlertid arbeidskrevende å lage analyse-øvinger som både er ikke-trivielle, interessante og rimelig fri for snubletråder. Det ville vært sterkt ønskelig med mulighet til mer *mengdetrening* for å repetere og bedre retensjon av begrepsapparat og tekniske ferdigheter.

Jeg lagde en prototype på autogenerert mengdetrening (Figur 8), tilgjengelig på https://jonovik.shinyapps.io/ggplot-sparring. Med utgangspunkt i en dataramme over

søvnbehov og levesett hos ulike dyrearter, lagde jeg et dataprogram som tilfeldig valgte to variabler og ba studenten plotte dem mot hverandre. Vanskelighetsgraden kan varieres ved å slå av/på visning av plottet og/eller programkoden, og ved å beskrive variablene ved ord istedenfor navn (Figur 9). (Ideelt sett skulle jeg utvidet dette til å velge hensiktsmessige grafiske elementer for ulike problemstillinger, men dét er mye vanskeligere å formalisere.)



Figur 8. En webside som lager tilfeldige oppgaver i plotting. Se neste figur for hva som skjer når studenten trykker på knappen.



Figur 9. Tilfeldig generert oppgave, med "vis løsningsforslag" delvis påslått. Vanskelighetsgraden er tatt et hakk opp ved at studenten selv må skjønne at "hvor mange timer i døgnet som dyret er våken" svarer til variabelen awake*, osv.*

Neste år ønsker jeg å bruke case-metodikkens tre dimensjoner (Pettersen 2005, 177) mer systematisk i øvingsutformingen, med gjennomtenkte variasjonsmuligheter langs hver dimensjon: analytisk, begreps- og presentasjonsmessig. Jeg håper det vil spenne ut et bedre rom av utfordringer for studenter på ulike mestringsnivå, og gi flere alternative og passe bratte læringskurver. Jeg ser i ettertid at årets øvinger har vært uhyre enkle i sin presentasjon for ikke å forvirre studentene. Kanskje burde vi tilby mer utfordringer i å gjenkjenne det viktigste blant mye tilgjengelig informasjon. (Dette ble dog rikelig ivaretatt i eksamensoppgaven, ifølge tilbakemeldingene vi fikk.) Den analytiske dimensjonen startet enkelt og steg jevnt utover, men burde kanskje beholdt enklere innpåmarsj i hver nye øving. Begrepsdimensjonen har vært holdt fokusert, men åpen. Det er med hensikt: Vi ønsker at studentene skal få en *moden* forståelse av noen få kjernesett av begreper.

Kursopplegget syntes å støtte studenter mot å gi opp ved motgang. Det er bra, fordi et slikt tankemønster kjennetegner studenter som står i fare for å dette av lasset i innlæring av programmering (Robins, Rountree, and Rountree 2003). Vi unngikk dette i stor grad gjennom flere bevisste tiltak. Live koding der foreleseren taster feil og tabber seg ut, bidrar til å

ufarliggjøre prøving og feiling. Øvingslærerne og foreleserne var mye tilgjengelige for studentene, og satte seg entusiastisk ned og snakket om hindrene studentene sto ovenfor. Vi hadde vel fire studenter som var sterkt i tvil om de skulle våge å fortsette, som fullførte etter personlig veiledning. Jeg tror STIN100 bygger en tiltro til egen evne til dataanalyse som vil komme dem til nytte i senere masteroppgaver og arbeidsliv.

Dette året har vi nok vært priviligierte idet vi kun har hatt studenter som aktivt har valgt kurset. Det kan by på ganske andre utfordringer å utløse læring hos folk som egentlig ikke vil, slik som noen i grunnkurset i statistikk. Kanskje det vil hjelpe å tydeligere formulere grader av måloppnåelse, så folk kan ta et informert valg av ambisjonsnivå. På den positive siden er jo læringsmålene for kurset ferdigheter som vil lette arbeidet i alle senere kurs og semesteroppgaver der de analyserer data. Målet om "ikke noe oversiktsstoff" bidrar forhåpentligvis til at selv matte- og datavegrerne føler de får valuta for investert tid, og at denne tida brukes til aktiv læring som sitter til praktisk bruk.

En utfordring jeg ikke hadde tenkt over, er arbeidsmengden ved evaluering av studentbesvarelser. Dybdeøvinger som krever refleksjon og individuelle løsninger, krever mye mer tid å evaluere og gi tilbakemelding på enn f.eks. multiple choice-svar. Utfordringer blir desto større når kurset skal skaleres opp. En kompromissløsning kan være godt begrunnede multiple choice-oppgaver, etter mønster av kasusbeskrivelsene i medisin- og veterinærutdanningen. Slike presenterer en pasient med symptomer, og spør f.eks. hvilke undersøkelser som bør gjøres, hvilken lidelse pasienten kan ha, eller hvilken behandling som bør iverksettes. På lignende måte kunne vi beskrevet et datasett og stilt et biologisk spørsmål, og spurt hvordan man bør utforme en datavisualisering for å belyse dette, eventuelt hvilke biologiske tolkninger som har støtte i en gitt figur, eller hvilke(n) av flere figurer som kommuniserer datasettet mest sannferdig og effektivt.

5. Utfordringer framover: Hvordan skalere til flere studenter, og hvilke muligheter åpner STIN100 for andre kurs?

Med rundt femti studenter går det såvidt an å ha en personlig relasjon til hver student, kunne navnene deres og huske hva de liker og hva de strever med. Dette blir vanskeligere hvis kurset blir obligatorisk og vokser til hundre studenter. En mulig strategi er å ha flere øvingslærere og delegere mer langsgående veiledning til dem, f.eks. ved å gjøre hver øvingslærer til "fadder" for en samling par.

Det øvrige kurstilbudet på fakultetet vil dra nytte av at studentene har grunnferdigheter i dataanalyse og reproduserbar rapportering. Kommende vår skal jeg f.eks. samarbeide med kursansvarlig for mikrobiell økologi om å få STIN100-aktige ting inn i deres labrapportering. Da blir det f.eks. lett å plotte hver gruppes resultater med hele klassens tall som bakteppe.

Til neste år vurderer jeg å legge tydeligere struktur på parprogrammeringen: Fordele roller (sjåfør vs navigatør) og angi tider for rollebytte. Jeg tror det kan gi enda raskere læring og

dessuten unngå en del tilfeller der én programmerte mens en annen skrev tekst. Videre vurderer jeg å bytte par mellom hver øvingsuke for å knytte flere sosiale bånd i klassen, spre erfaringer og arbeidsmåter bedre, og fordele belastningen ved eventuelle latsabber jevnere utover. (To grupper meldte om latsabb på laget i høst.)

I sum vil jeg si at dette kurset nok er det morsomste jeg har holdt, takket være dyktige og morsomme kollegaer og engasjerte og positive studenter. Ekstra artig er det at det falt sammen i tid med pedagogikk-kurset. Det har gitt mange nye perspektiver som jeg skal bruke i forbedringen av kurset til neste år.

6. Referanser

Biggs, John. 2011. *Teaching For Quality Learning At University*. 4 edition. Open University Press.

Bloom, Benjamin S. 1956. *Taxonomy of Educational Objectives, Handbook 1: Cognitive Domain*. 2nd edition Edition edition. New York: Addison-Wesley Longman Ltd.

Brown, Neil C. C., and Greg Wilson. 2018. "Ten Quick Tips for Teaching Programming." *PLOS Computational Biology* 14 (4): e1006023. https://doi.org/10.1371/journal.pcbi.1006023.

CNN. 1986. *Challenger Disaster Live on CNN*. https://www.youtube.com/watch?v=j4JOjcDFtBE.

Gandrud, Christopher. 2015. *Reproducible Research with R and R Studio, Second Edition*. 2 edition. Boca Raton: Routledge.

Johnson, David W., Roger T. Johnson, and Karl A. Smith. 2014. "Cooperative Learning: Improving University Instruction by Basing Practice on Validated Theory." *Journal on Excellence in College Teaching* 25: 85–118.

Knuth, D. E. 1984. "Literate Programming." *The Computer Journal* 27 (2): 97–111. https://doi.org/10.1093/comjnl/27.2.97.

Lui, Kim Man, and Keith C.C. Chan. 2006. "Pair Programming Productivity: Novice–Novice vs. Expert–Expert." *International Journal of Human-Computer Studies* 64 (9): 915–25. https://doi.org/10.1016/j.ijhcs.2006.04.010.

McDowell, Charlie, Linda Werner, Heather E. Bullock, and Julian Fernald. 2006. "Pair Programming Improves Student Retention, Confidence, and Program Quality." *Communications of the Association for Computing Machinery* 49 (8): 90–95. https://doi.org/10.1145/1145287.1145293.

NMBUs studienemnd. 2015. "NMBU læringsfilosofi." June 25, 2015. https://www.nmbu.no/ffu/filosofi.

Peng, Roger D. 2011. "Reproducible Research in Computational Science." *Science* 334 (6060): 1226–27. https://doi.org/10.1126/science.1213847.

Pettersen, Roar C. 2005. *Kvalitetslæring i høgere utdanning: Innføring i problem- og praksisbasert didaktikk*. Oslo: Universitetsforlaget.

http://www.universitetsforlaget.no/nettbutikk/kvalitetslaering-i-hoyere-utdanning-uf.html.

Robins, Anthony, Janet Rountree, and Nathan Rountree. 2003. “Learning and Teaching Programming: A Review and Discussion.” *Computer Science Education* 13 (2): 137–72. https://doi.org/10.1076/csed.13.2.137.14200.

Rubin, Marc J. 2013. “The Effectiveness of Live-Coding to Teach Introductory Programming.” In *Proceeding of the 44th ACM Technical Symposium on Computer Science Education*, 651–656. SIGCSE ’13. New York, NY, USA: ACM. https://doi.org/10.1145/2445196.2445388.

Tufte, Edward R. 1997. *Visual Explanations: Images and Quantities, Evidence and Narrative*. Graphics Press.

Vygotskij, Lev Semenovič. 1934. *Tenkning og tale*. Edited by Alex Kozulin. Basis. Oslo: Gyldendal akademisk.

Wickham, Hadley. 2009. *Ggplot2: Elegant Graphics for Data Analysis*. 1st ed. New York: Springer.

Wickham, Hadley, and Garrett Grolemund. 2016. *R for Data Science: Import, Tidy, Transform, Visualize, and Model Data*. 1 edition. O’Reilly Media. https://r4ds.had.co.nz/.

Williams, L., R. R. Kessler, W. Cunningham, and R. Jeffries. 2000. “Strengthening the Case for Pair Programming.” *IEEE Software* 17 (4): 19–25. https://doi.org/10.1109/52.854064.

Xie, Yihui. 2015. *Dynamic Documents with R and Knitr, Second Edition*. 2 edition. Boca Raton: Routledge.