R fyrir byrjendur

Frá tilraun til gagna

Anna Helga Jónsdóttir
Bjarki Þór Elvarsson
Sigrún Helga Lund

Skráning gagna

1. Hefur hver breyta sinn dálk og hvert viðfangsefni sína línu?
2. Eru nokkrar óþarfa línur?
3. Inniheldur skráin sérístenska stafi?
4. Eru nokkur bil?
5. Eru sömu útkomurnar skráðar á sama hátt?
6. Hefur hver breyta sitt nafn?
7. Er fullt samræmi í því hvernig tölur eru skráðar?
8. Eru nokkrir punktar sem gætu valið misskilningi?

Safn talna - vigrar

Vigur getur innihaldið

- tölur - integer, double (numeric)
- bókstafi - character
- röktákn - logical

```
x <- c(1,2,3)
y <- c("a","b","c")
z1 <- c(TRUE,FALSE,TRUE)
z2 <- c(T,F,T) # sama
class(x) # skilar gerð vigurs

## [1] "numeric"
```

Nokkrar gagnlegar skipanir

```
c(1,2)            # býr til vigur með stökunum 1 og 2
seq(1,10)         # býr til vigur frá 1 til 10
1:10              # sama
rep(2,3)          # býr til vigur sem inniheldur 2,
                    # þrisvar
length(vigur)     # skilar lengdinni á vigur
sort(vigur)       # raðar stökum í vigur eftir röð
rank(vigur)       # hvar í stærðarröð er stakið
order(vigur)      # skilar vísi á stærðarröð
cbind()           # bindur saman vigra/gagnatoflur
                    # á dálkum
rbind()           # bindur saman vigra/gagnatoflur
                    # á röðum
```

Vísa í stak/stök vigurs

```
x1 <- c(7,3,9,2,1,5)
x1[1]         # ná í fyrsta stak x1
x1[c(1,2)]    # ná í fyrsta og annað stak
x1[-1]        # ná í öll stök nema það fyrsta
x1[-c(1,2)]   # má í öll stök nema fyrsta og annað

which(x1==9) # skilar vísi þar sem
             # skilyrðið er uppfyllt
```

Gagnatöflur - dataframe

- Þegar við vinnum með gagnasöfn með fleiri en einni breytu er gott að nota gagnatöflur (dataframe)
- Dálkar í gagnatöflu þurfa ekki að vera af sömu gerð
- Séu gögn lesin inn í R með t.d. `read.table()` eða `read.csv()` eru þau geymd sem gagnatafla
- Við notum $ til að vísa í ákveðna breytu í gagnatöflu. Til að ná í breytuna `breyta` í gagnatöflu sem ber nafnið `dat` gerum við það með

```
dat$breyta
```

Gögn lesin inn með read.table()

- `read.table()` og `read.csv()` eru svipaðar, svolítill munur á sjálfgefnu stillingunum (argument)
- Skila báðar gagnatöflu (dataframe)
- Hefur fjöldann allan af stillingum/rofum, gott að lesa `help(read.table)` vel - getur sparað mikinn tíma
- Svolítið gamaldags...

Nokkrar nytsamlegar stillingar í `read.table()`

- `header`: innihaldi efsta lína í skráinn breytunöfn skal setja `header = TRUE`
- `sep`: segir til um hvað er notað til að aðgreina dálka/breytur í skránni
- `dec`: segir til um hvað er notað til að aðskilja heiltöluhluta og tugabrot
- `na.strings`: segir til um hvernig NA gildi eru kóðuð í skránni
- `nrows`: segir til um hámarks fjölda raða sem lesa á inn
- `skip`: segir til um hversu margar línur á að "hoppa yfir"
- `comment.char`: segir til um hvað er notað til að tákna komment í skránni
- `encoding`: Segir til um hvaða stafagerð er notuð í skránni

Gögn lesin inn með `read.table()`

Viljum við lesa inn skrána `gogn.csv` sem

- inniheldur breytunöfn í efstu línunni
- ";" er notað til að aðgreina breytur
- "," er notað til að aðgreina heiltöluhluta og tugabrot (gaaaarg)
- NA gildi eru kóðuð með "#N/A"

og geyma sem dataframe að nafni `dat` gerum við það með:

```
dat <- read.table("gogn.csv",sep=";",dec=",",header=T,
                  na.strings="#N/A")
```

Töflungar - tibbles

- Töflungar er nútímaleg útfærsla á gagnatöflum
- Tilheyrir `tibbles` pakkanum sem er hluti af `tidyverse`
- Haga sér á svipaðann hátt og gagnatöflur en helsti munur er hvernig innihaldið er sýnt
- Við notum $ til að vísa í ákveðna breytu í töflungi. Til að ná í breytuna `breyta` í töflungi sem ber nafnið `dat` gerum við það með

```
dat$breyta
```

Gögn lesin inn með `read_delim()`

- Aðferðirnar `read_delim()` og `read_csv()` hafa tekið við af `read.table()` og `read.csv()`
- Tilheyra `readr` pakkanum sem er hluti af `tidyverse`
- Skila töflungi, ekki gagnatöflu

Gögn lesin inn með read_delim()

- `read_delim()` og `read_csv()` svipaðar, svolítill munur á sjálfgefnu stillingunum
- Skila báðar töflungi (tibble)
- Hefur fjöldann allan af stillingum, gott að lesa `help(read_delim)` vel - getur sparað mikinn tíma
- Ef "," er notuð til að aðgreina heiltöluhluta og tugabrot skoðið þá `read_csv2()`

Gögn lesin inn með `read_delim()`

Viljum við lesa inn skrána `gogn.csv` sem

- inniheldur breytunöfn í efstu línunni
- ";" er notað til að aðgreina breytur
- NA gildi eru kóðuð með "#N/A"

og geyma sem dataframe að nafni `dat` gerum við það með:

```
dat <- read_delim("gogn.csv",delim=";",na="#N/A")
```

Annars konar gögn lesin inn

- Lesa má Excel skrár (`.xls` og `.xlsx` skrár) með `read_excel()` aðferðinni sem er hluti af `readxl` pakkanum. Hann er hluti af `tidiverse` en þó þarf að hlaða honum sérstaklega inn með `library(readxl)`
- Lesa má SPSS skrár (`.sav`) inn með `read_spss()` aðferðinni sem er hluti af `haven` pakkanum. Hann er hluti af `tidiverse` en þó þarf að hlaða honum sérstaklega inn með `library(haven)`
- Báðar aðferðirnar skila töflungi
- Fjöldinn allur af öðrum aðferðum - googlið!

Gróft yfirlit gagna

```r
head(dat)          # sýnir efstu sex línurnar í dat
dim(dat)           # skilar fjölda lína og dálka í dat
names(dat)         # skilar nöfn breyta í dat
glimpse(dat)       # skilar gerð breyta í dat
```

Gögn skrifuð í skrá

Sams konar aðferðir til að skrifa gögn í skrá:

```
write_csv()    # komma notud sem adgreinir
write_delim()  # haegt ad tilgreina adgreini
write_tsv()    # tab notad sem adgreinir
write_excel_csv()  # komma notud sem adgreinir, til ad lesa i Excel
```

Vöntun mælinga

- Mælingar sem vantar eru táknaðar með `NA` í R
- Til eru ýmsar aðferðir til að taka á vöntun mælinga
- `is.na()`: skilar `T` eða `F` fyrir hvert stak í vigrinum/fylkinu/datafame
- `na.omit()`: fjarlægir þær raðir sem gildi vantar í

Gögn lesin inn - pulsAll.csv

Breyta	Útskýring/gildi
namskeid	LAN203, STAE209
kronukast	thorskur, landvaettir
haed	hæð í sentimetrum
thyngd	þyngd í kílógrömmum
aldur	aldur í árum
kyn	1 = kona, 2 = karl, 3 = annad
reykir	ja, nei
drekkur	ja, nei
likamsraekt	líkamsrækt í klst/viku
fyrriPuls	fyrri púlsmæling slög/mín
seinniPuls	seinni púlsmæling slög/mín
inngrip	hljop, sat_kyrr
dagsetning	hvaða dag tilraunin fór fram dagur.manudur.ar

read_delim() - gerð breyta

```
library(tidyverse)

puls <- read_delim("pulsAll.csv",delim=";")
```

```
## Parsed with column specification:
## cols(
##  namskeid = col_character(),
##  kronukast = col_character(),
##  haed = col_double(),
##  thyngd = col_double(),
##  aldur = col_integer(),
##  kyn = col_integer(),
##  reykir = col_character(),
##  drekkur = col_character(),
##  likamsraekt = col_double(),
##  fyrriPuls = col_integer(),
##  seinniPuls = col_integer(),
##  inngrip = col_character(),
##  dagsetning = col_character()
```

Ákveðin gildi valin úr gagnatöflu/töflungi

- Getum notað [ ] eins og við sáum fyrir vigra: dat[visar i linur, visar i dalka]
- Getum notað samanburðarvirkja í [ ]: ==, !=, >, <,...
- %in%
- Aðferðir úr `dplyr` pakkanum: `slice()`, `select()`, `filter()`, ...

Ákveðin gildi valin úr töflungi (tibbles)

Búum til nýjan töflung sem inniheldur alla nemendur í námskeiðinu STÆ209:

```
d.stae <- filter(puls, namskeid=="STAE209")
```

Búum til nýjan töflung sem inniheldur alla nemendur í STÆ209 sem eru hærri en 170 cm á hæð:

```
d.stae.ha <- filter(puls, namskeid=="STAE209", haed>170)
```

Búum til nýjan töflung með nemendum sem eru 20, 22 og 24 ára:

```
aldur.nota <- c(20,22,24)
d.ald <- filter(puls, aldur%in%aldur.nota)
```

Kóðun breyta

Þegar búið er að lesa gögn inn þurfum við oft að

1. búa til nýjar breytur
2. kóða breytur sem voru skráðar sem tölur í gögnunum okkar sem flokkabreytur og nefna flokkanna
3. skipta talnabreytum upp í bil og búa til flokkabreytur út frá þeim
4. sameina tvo eða fleiri flokka í flokkabreytu.
5. ...

Bæta við eða yfirskrifa breytu í töflungi

- Notum `mutate()` aðferðina til að búa til nýja breytu eða yfirskrifa breytu sem nú þegar er til í töflungnum
- `mutate()` tilheyrir `dplyr` pakkanum sem er hluti af `tidyverse`
- Búum til nýja breytu, `bmi` sem innheldur BMI gildi einstaklinganna í `puls` gagnasafninu:

```
puls <- mutate(puls, bmi = thyngd/(haed/100)^2)
```

Flokkabreytur í R

- Faktorar (factors) eru notaðir til að vinna með flokkabreytur en það eru breytur sem geta tekið fyrirfram skilgreind gildi, flokka
- Mörg af eldri föllum í R breyta stafabreytum (characters) í flokkabreytur en þau nýrri gera það ekki
- Í `forcats` pakkanum má finna ýmsar aðferðir til að vinna með flokkabreytur. Hann er hluti af `tidiverse` en þó þarf að hlaða honum sérstaklega inn með `library(forcats)`

Talnabreytum breytt í flokkabreytur

- Oft eru notaðar tölur til að aðgreina flokka í breytu.
- Þegar slík breyta er lesin inn í R er hún lesin inn sem talnabreyta og þurfum við því að breyta henni í flokkabreytu eftir innlesturinn eða búa til nýja breytu
- Til þess getum við notað `factor()` aðferðina.

```
puls <- mutate(puls, kyn = factor(kyn))
```

- Ég mæli þó með að skilgreina flokkana fyrirfram og nota `parse_factor()` aðferðina

```
kyn.f <- c("1","2")
puls <- mutate(puls, kyn = parse_factor(kyn, levels=kyn.f))
```

Breyta nöfnum á flokkum

- Viljum oft breyta nöfnunum á flokkum flokkabreyta
- Notum `fct_recode()` aðferðina.

Getum byrjað á því að kanna hvað flokkarnir heita:

```
levels(puls$kyn)

## [1] "1" "2"
```

og breytum svo:

```
puls <- mutate(puls, kyn = fct_recode(kyn, kvk = "1", kk = "2"))
```

Skipta talnabreytu upp í flokka

- Stundum viljum við skipta gildum talnabreytu upp í flokka.
- notum `cut()` aðferðina.
- Búum til nýja breytu, `haed.flokkur` þar sem breytunni `haed` er skipt upp í eftirfarandi flokka: `[0,160)`, `[160,180)`, `[180,210)` og nefnum flokkana upp á nýtt.

Skipta talnabreytu upp í flokka

```
puls <- mutate(puls, haed.flokkur = cut(haed,
                    c(0,160,180,210), right=F))

levels(puls$haed.flokkur)

## [1] "[0,160)"   "[160,180)" "[180,210)"

puls <- mutate(puls, haed.flokkur =
       fct_recode(haed.flokkur,
       lágvaxin = "[0,160)", miðlungs = "[160,180)",
       hávaxin = "[180,210)"))
```

Breyta röðun á flokkum flokkabreytu

Núverandi röðun á flokkunum er:

```
levels(puls$haed.flokkur)

## [1] "lágvaxin" "miðlungs" "hávaxin"
```

Þurfum stundum að breyta röðun, t.d. áður en við gerum myndir:

```
puls <- mutate(puls, haed.flokkur =
            parse_factor(haed.flokkur,
            levels=c("hávaxin", "miðlungs", "lágvaxin")))
levels(puls$haed.flokkur)

## [1] "hávaxin"  "miðlungs" "lágvaxin" NA
```

Sameina tvo eða fleiri flokka flokkabreytu

Viljum stundum sameina tvo eða fleiri flokka flokkabreytu:

Sameinum `lágvaxin` og `miðlungs` í flokkinn `lægri` og skýrum hinn flokkinn `hærri`:

```
puls <- mutate(puls, haed.flokkur.s =
        fct_recode(haed.flokkur,
        lægri = "lágvaxin", lægri = "miðlungs", hærri = "hávaxin"))
levels(puls$haed.flokkur.s)

## [1] "hærri" "lægri" NA
```

Þetta er bara byrjunin...

- Aðferðirnar sem við höfum séð hér eru aðeins nokkrar af fjöldanum öllum af nytsamlegum aðferðum sem finna má í `dplyr` og fleiri pökkum
- t.d má nota
  - `arrange()` til að endurraða röðum í gagnatöflu
  - `gather()`, `shape()`, `seperate()`, `unite()` til að breyta lögun gagnanna okkar
  - ...
- Googlið!

Pípuriháttur %>%

- Þegar skipanirnar okkar fara að verða flóknari kemur sér vel að nota pípurithátt, %>%
- Pípan, %>%, tekur það sem er vinstra megin og notar það sem fyrstu stillinguna í aðferðinni sem er henni á hægri hönd
- Eftirfarandi skipanir gera það sama:

```
puls <- mutate(puls, kyn = factor(kyn))

puls %>% mutate(kyn = factor(kyn)) -> puls
```