Introduktion

Min R-kode er vedlagt som appendix til projektet.

1 Opgave 1

1.1 Delopgave a

Reglen for differentiering af sammensatte funktioner giver os:

$$\frac{du}{dt} = \frac{1}{K^\alpha}\alpha y^{\alpha-1}\frac{dy}{dt} \tag{1}$$

Ved at regne videre på dette udtryk får vi

$$\frac{du}{dt} = \frac{\alpha y^\alpha}{K^\alpha}y^{-1}\frac{dy}{dt} = \frac{\alpha\frac{y^\alpha}{K^\alpha}}{y}\frac{dy}{dt} = \frac{\alpha u}{y}\frac{dy}{dt} \tag{2}$$

1.2 Delopgave b

Fra linje (2) får vi let et udtryk for y differentieret med hensyn til t:

$$\frac{dy}{dt} = \frac{y}{\alpha u}\frac{du}{dt} \tag{3}$$

Differentialligningen for y giver os et andet udtryk for y differentieret med hensyn til t. Vi kan altså sætte disse to udtryk lig hinanden:

$$\frac{y}{\alpha u}\frac{du}{dt} = ry\left(1 - \left(\frac{y}{K}\right)^\alpha\right) \tag{4}$$

hvilket er ækvivalent med

$$\frac{du}{dt} = \alpha ru\left(1 - \left(\frac{y}{K}\right)^\alpha\right) \tag{5}$$

hvilket per definition af u er ækvivalent med

$$\frac{du}{dt} = \alpha ru\left(1 - u\right) \tag{6}$$

1.3 Delopgave c

Linje (6) er ækvivalent med

$$\frac{du}{dt} = (\alpha r)u\left(1 - \frac{u}{1}\right) \tag{7}$$

Her har vi altså en helt normalt logistisk differentialligning med vækstrate αr og befolkningskapacitet 1. Vi ved, at den fuldstændige løsning er

$$u(t) = \frac{1}{1 + c\exp(-\alpha r t)}, \quad c \in \mathbb{R} \tag{8}$$

1.4 Delopgave d

Per definition af u følger, at

$$y = Ku^{\frac{1}{\alpha}} \tag{9}$$

Altså er

$$y(t) = K(u(t))^{\frac{1}{\alpha}} = K\left(\frac{1}{1 + c\exp(-\alpha r t)}\right)^{\frac{1}{\alpha}} = \frac{K}{(1 + c\exp(-\alpha r t))^{\frac{1}{\alpha}}} \tag{10}$$

1.5 Delopgave e

Vi ser, at $y(t) = 0$ og $y(t) = K$ begge er konstante løsninger til differentialligningen for y (en konstant funktion differentieret er altid 0):

$$r \cdot 0\left(1 - \left(\frac{0}{K}\right)^{\alpha}\right) = 0 \tag{11}$$

$$\tag{12}$$

$$r \cdot K\left(1 - \left(\frac{K}{K}\right)^{\alpha}\right) = r \cdot K\,(1 - 1) = 0 \tag{13}$$

Altså er $y^* = 0$ og $y^* = K$ begge ligevægte for differentialligningen.

Jeg vil nu bruge sætning 9 på side 150 i kursusbogen Differentialligninger til at afgøre om ligevægtene er lokalt stabile eller ej.

Da

$$f(y) = ry\left(1 - \left(\frac{y}{K}\right)^{\alpha}\right) = r\left(y - \frac{y^{\alpha+1}}{K^{\alpha}}\right) \tag{14}$$

afbilleder fra $\mathbb{R}$ til $\mathbb{R}$ er dens funktionalmatrix simpelthen blot dens afledede med hensyn til y:

$$f'(y) = r\left(1 - \frac{\alpha y^{\alpha}}{K^{\alpha}}\right) = r\left(1 - \alpha\left(\frac{y}{K}\right)^{\alpha}\right) \tag{15}$$

Vi evaluerer dette udtryk i ligevægtene

$$f'(0) = r\left(1 - \alpha\left(\frac{0}{K}\right)^{\alpha}\right) = r \tag{16}$$

$$\tag{17}$$

$$f'(K) = r\left(1 - \alpha\left(\frac{K}{K}\right)^{\alpha}\right) = -r\alpha \tag{18}$$

Vi kan betragte konstanterne r og $-r\alpha$ som 1×1 matricer. Da en 1×1 matrix

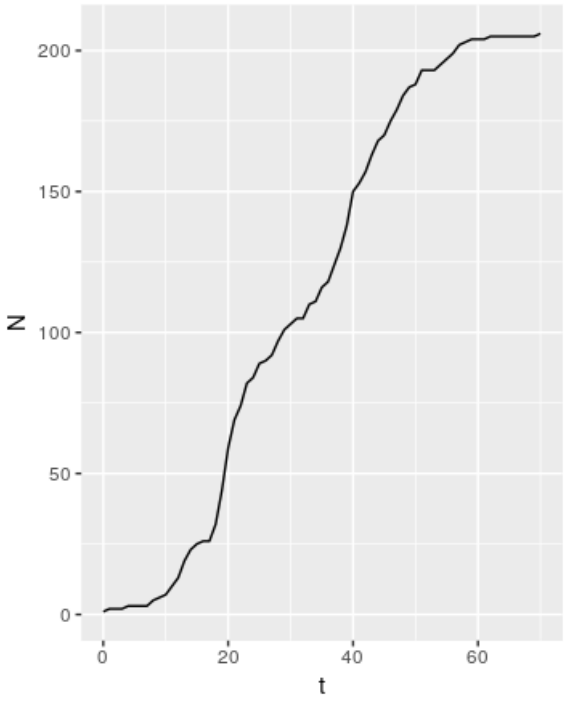
$$[a] \tag{19}$$

altid har a som sin egenværdi, har funktionalmatricen evauleret i 0 og K altså henholdsvis r og $-r\alpha$ som egenværdi. Da r og α begge er skarpt større end 0, så har funktionalmatricen evauleret i 0 og K altså henholdsvis en ikke-negativ og en negativ egenværdi. Altså er K en lokalt stabil ligevægt, hvorimod 0 ikke er lokalt stabil.

2 Opgave 2

2.1 Delopgave a

Her er et plot over antal smittede N som funktion af dagen t:

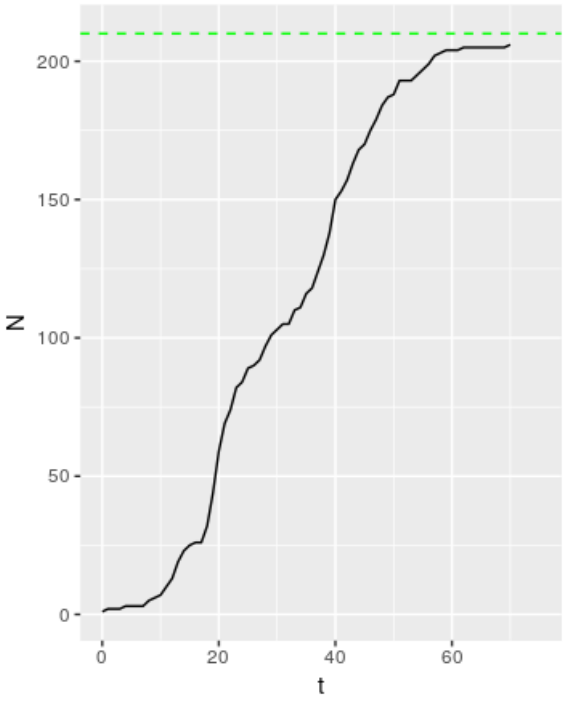


2.2 Delopgave b

1. Den fuldstændige løsning til den logistisk differentialligning er givet som

$$N(t) = \frac{K}{1 + c\exp(-rt)} \tag{20}$$

2. Ud fra de sidste målinger, sætter jeg $K = 208$. Vi ved, at en logistisk funktion konvergerer mod, men aldrig når K. Jeg har derfor sat K lidt større end de sidste målinger, da vi godt nok kan se i data, at vækstraten er aftagende, men jeg synes det virker mest rimeligt stadig at gemme lidt rum til yderligere vækst. Her er et plot, hvor K er tegnet ind:



Hermed får vi, at vores model er på formen

$$N(t) = \frac{208}{1 + c\exp(-rt)} \tag{21}$$

3. Jeg antager, at $N(0) = 1$, da der i dataen er 1 smittet på den 0'te dag. Hermed får vi, at

$$N(0) = \frac{208}{1 + c\exp(-r \cdot 0)} = \frac{208}{1 + c} = 1 \tag{22}$$

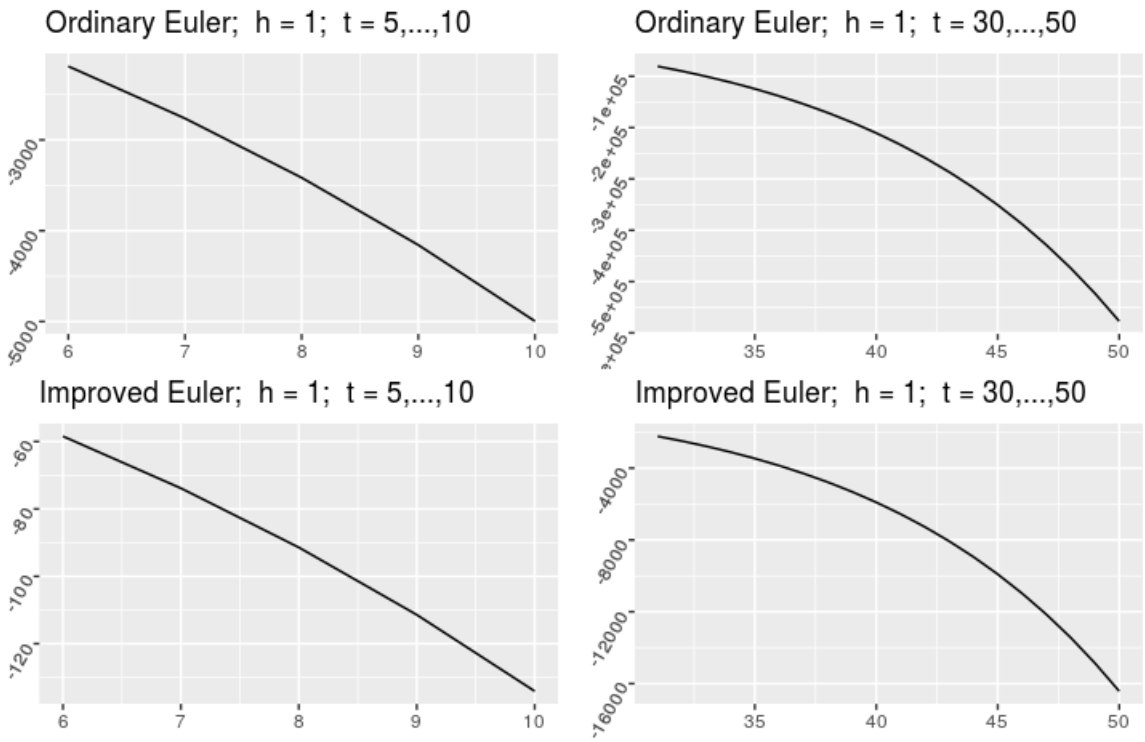
hvilket er ækvivalent med

$$c = 207 \tag{23}$$

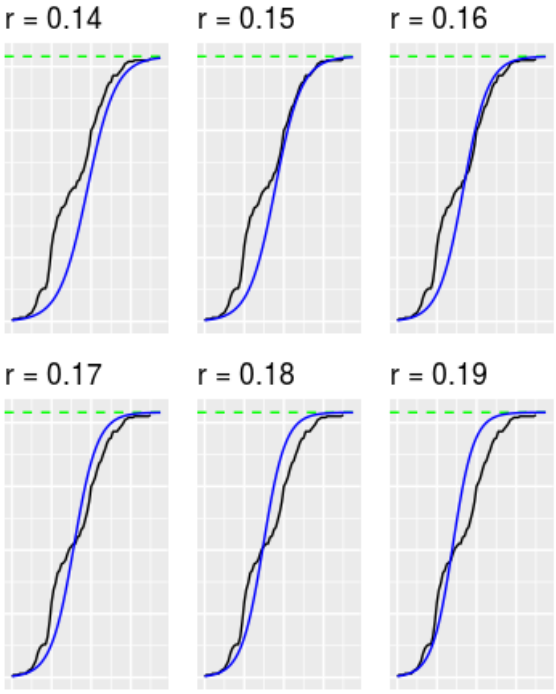
Hermed får vi, at vores model er på formen

$$N(t) = \frac{208}{1 + 207\exp(-rt)} \tag{24}$$

4. Jeg plotter først nogle eksempler på modeller med r mellem 0.12 og 0.22:



Ud fra disse plots vælger jeg at lave nogle plots, der ser nærmere på modeller med r mellem 0.14 og 0.19:



Ud fra disse plots vælger jeg at sætte $r = 0.17$, da denne model er nogenlunde præcis og desuden er den model, der ser ud til at fordele sine residualer mest ligeligt udover alle dagene.

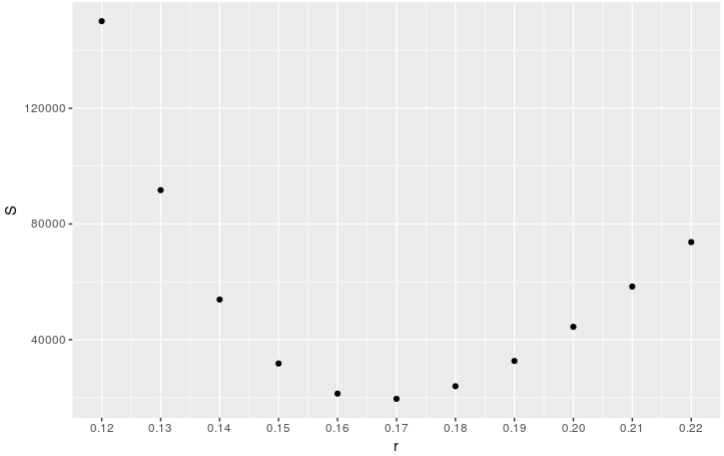
5. Min endelige model er altså

$$N(t) = \frac{208}{1 + 207\exp(-0.17t)} \tag{25}$$

En standard måde at måle hvor godt en model passer på data er summen af modellens kvadrerede residualer, hvilket i vores tilfælde er givet som

$$S = \sum_{t=0}^{70} (N_i - N(t))^2 \tag{26}$$

hvor $N_0, N_1, ..., N_{70}$ er de faktiske optællinger af syge på dag $0, 1, ..., 70$. Min valgte model har $S = 19549.88$. Dette tal siger dog intet som helst i sig selv uden en eller anden form for kontekst. Her er et plot af S for $r = 0.12, 0.13, ..., 0.21, 0.22$:



Her ser vi, at hvis vi bruger S som mål for, hvor god vores model er, så er $r = 0.17$ det bedste valg ud af de testede værdier af r, givet mit valg af $K = 208$ og $N(0) = 1$.

2.3 Delopgave c

1. Den fuldstændige løsning til den modificerede logistisk differentialligning er givet som

$$N(t) = \frac{K}{(1 + c\exp(-\alpha rt))^{\frac{1}{\alpha}}} \tag{27}$$

2. Da $\alpha, r > 0$, har vi

$$\exp(-\alpha rt) \to 0, \quad t \to \infty \tag{28}$$

Heraf følger, at

$$1 + c\exp(-\alpha rt) \to 1, \quad t \to \infty \tag{29}$$

Heraf følger, at

$$(1 + c\exp(-\alpha rt))^{\frac{1}{\alpha}} \to 1, \quad t \to \infty \tag{30}$$

Heraf følger, at

$$N(t) = \frac{K}{(1 + c\exp(-\alpha rt))^{\frac{1}{\alpha}}} \to K, \quad t \to \infty \tag{31}$$

Ud fra de sidste målinger, sætter jeg som ovenfor $K = 208$. Hermed får vi, at vores model er på formen

$$N(t) = \frac{208}{(1 + c\exp(-\alpha r t))^{\frac{1}{\alpha}}} \tag{32}$$

3. Jeg antager, at $N(0) = 1$, da der i dataen er 1 smittet på den 0'te dag. Jeg antager desuden, at $\alpha = 5$. Hermed får vi, at

$$N(0) = \frac{208}{(1 + c\exp(-5r \cdot 0))^{\frac{1}{5}}} = \frac{208}{(1 + c)^{\frac{1}{5}}} = 1 \tag{33}$$

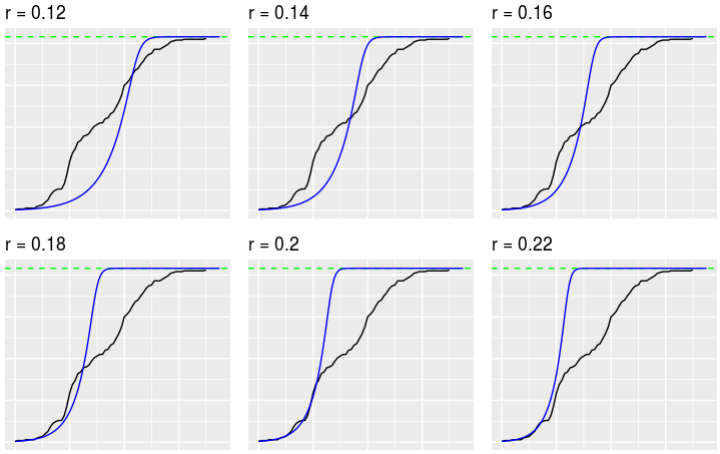
hvilket er ækvivalent med

$$c = 208^5 - 1 = 389328928767 \tag{34}$$

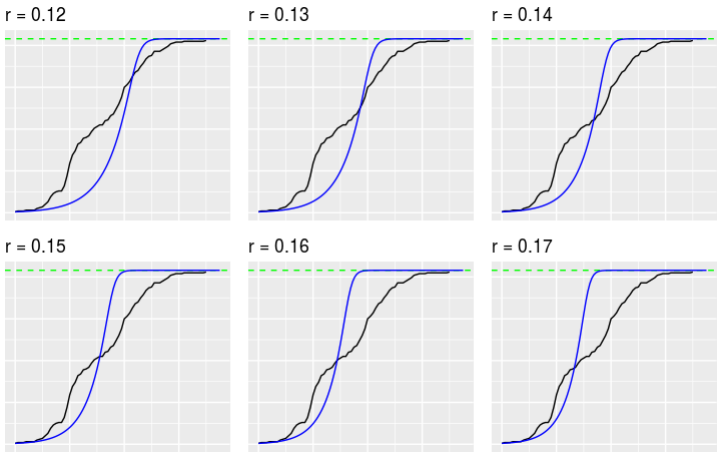
Hermed får vi, at vores model er på formen

$$N(t) = \frac{208}{(1 + 389328928767\exp(-5rt))^{\frac{1}{5}}} \tag{35}$$

Jeg plotter først nogle eksempler på modeller med r mellem 0.12 og 0.22:



Ud fra disse plots vælger jeg at lave nogle plots, der ser nærmere på modeller med r mellem 0.12 og 0.17:

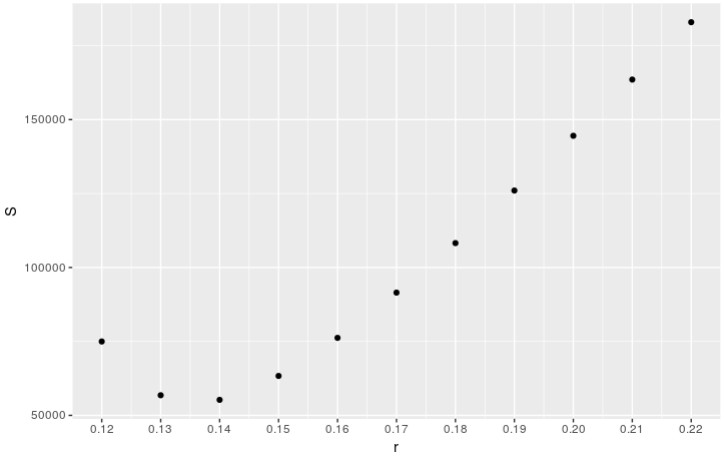


Ud fra disse plots vælger jeg at sætte $r = 0.14$.

Min endelige model er altså

$$N(t) = \frac{208}{(1 + 389328928767 \exp(-0.7t))^{\frac{1}{5}}} \tag{36}$$

Det er åbenlyst ud fra graferne, at denne model passer dårligere med data, end den simple logistiske model, jeg kom frem til i delopgave b. Her er et plot af summen af de kvadrerede residualer for $r = 0.12, 0.13, ..., 0.21, 0.22$:



Her ser vi, at hvis vi bruger S som mål for, hvor god vores model er, så er $r = 0.14$ det bedste valg ud af de testede værdier af r, givet valget af $K = 208$, $N(0) = 1$ og $\alpha = 5$. Modellen har i så fald $S = 55249.06$, hvilket er omtrent 2.5 gange så stort som S for modellen i delopgave b. Ud fra dette kriterium er modellen fra delopgave b altså også bedre.

4. Jeg antager, at $N(0) = 1$, da der i dataen er 1 smittet på den 0'te dag. Jeg antager desuden, at $\alpha = 0.2$. Hermed får vi, at

$$N(0) = \frac{208}{(1 + c\exp(-0.2r \cdot 0))^5} = \frac{208}{(1+c)^5} = 1 \tag{37}$$

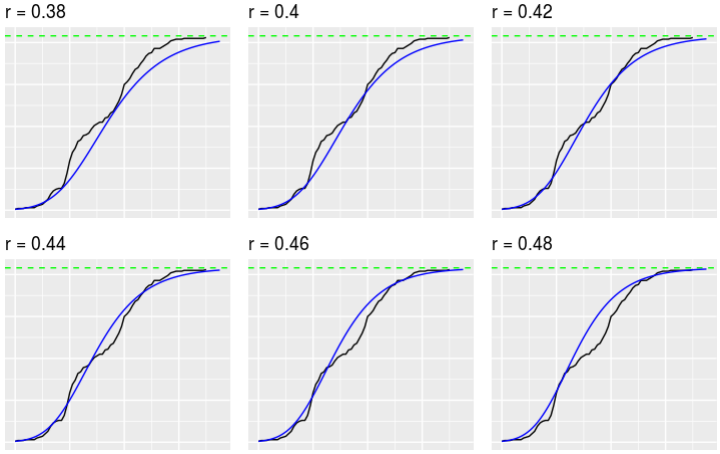
hvilket er ækvivalent med

$$c = 208^{\frac{1}{5}} - 1 = 1.908 \tag{38}$$

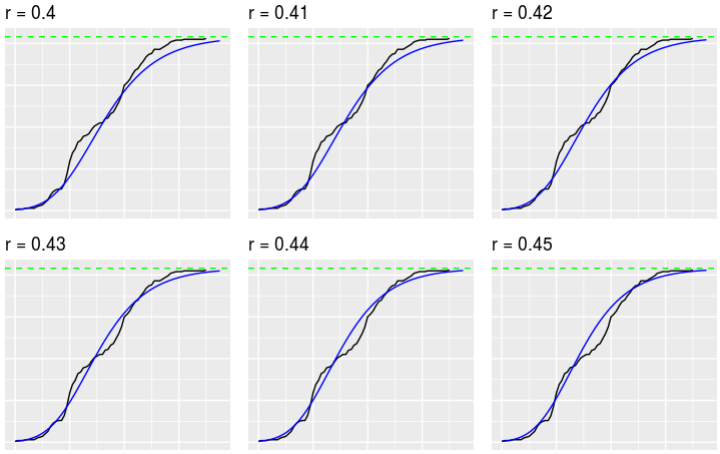
Hermed får vi, at vores model er på formen

$$N(t) = \frac{208}{(1 + 1.908\exp(-0.2rt))^5} \tag{39}$$

Jeg plotter først nogle eksempler på modeller med r mellem 0.38 og 0.48:



Ud fra disse plots vælger jeg at lave nogle plots, der ser nærmere på modeller med r mellem 0.4 og 0.45:



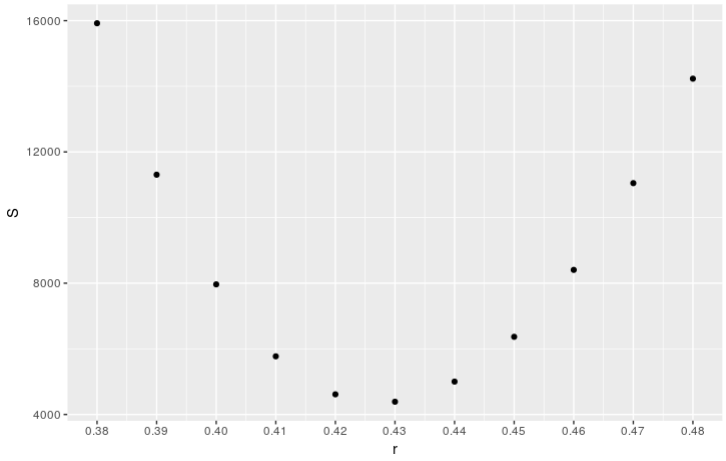
Ud fra disse plots vælger jeg at sætte $r = 0.43$.

Min endelige model er altså

$$N(t) = \frac{208}{(1 + 1.908 \exp(-0.82t))^5} \tag{40}$$

Det er ret klart ud fra graferne, at denne model passer bedre med data, end den simple logistiske model, jeg kom frem til i delopgave b. For det første er dens residualer mindre. For det andet - og lige så vigtigt - så begår den simple logistiske model en mere systematisk form for fejl, idet den konsekvent forudsiger for få antal syge i den første halvdel af epidemien og for mange syge i den sidste halvdel. I den nye model ser residualerne ud til at fordele sig mere tilfældigt omkring modellens forudsigelser (de ser dog ikke fuldstændigt tilfældige ud, men dog bedre end for den simple logistiske model).

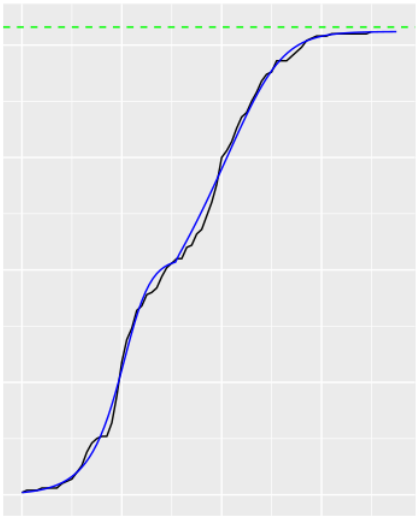
Her er et plot af summen af de kvadrerede residualer for $r = 0.38, 0.39, ..., 0.47, 0.48$ for den nye model:



Her ser vi, at hvis vi bruger S som mål for, hvor god vores model er, så er $r = 0.43$ det bedste valg ud af de testede værdier af r, givet valget af $K = 208$, $N(0) = 1$ og $\alpha = 0.2$. Modellen har i så fald $S = 4388.52$, hvilket er omtrent 5 gange så småt som S for modellen i delopgave b. Ud fra dette kriterium er den nye model altså også bedre end modellen fra delopgave b.

2.4 Delopgave d

1. Her er et plot for modellen af to kombinerede modificerede logistiske funktioner:



Da $N1(31) = 103.63$ og $N2(31) = 105$, har funktionen N ikke et særligt stort spring i $t = 31$.

2. Denne model er bestemt den af vores hidtidige modeller, der passer bedst på data. Det gælder både, når vi vurderer den ud fra grafen, og når vi ser, at den har $S = 667.29$, hvilket er det klart laveste hidtil. Dog ville jeg fra et metodisk perspektiv umiddelbart være skeptisk over for den måde, vi har fundet frem til modellen på. Basalt set har vi set på data, lagt mærke til et lidt mærkeligt bump og derefter konstrueret vores model ud fra dette bump. Hermed risikerer vi at overfitte til data og få en kompliceret model, der både er sværere at fortolke og har værre forudsigelseskraft på ny data, idet den har den foreliggende datas tilfældigheder bygget ind i sig.

3 Opgave 3

3.1 Delopgave a

3.1.1 (i)

Vi har den inhomogene, linære 1. ordens differentialligning

$$x'(t) = r_0 x(t) - u_0 - \alpha t \tag{41}$$

som vi kan omskrive til

$$\frac{dx}{dt} + (-r_0 x) = -u_0 - \alpha t \tag{42}$$

hvorved den er på samme form, som kursusbogen Matematik for Biovidenskab opskriver inhomogene, linære 1. ordens differentialligninger. Hermed kan vi bruge panserformlen på side 281 til at bestemme den fuldstændige løsning til differentialligningen

$$x(t) = \exp(-F(t)) \int \exp(F(t))g(t)\ dt \tag{43}$$

hvor

$$F(t) = \int -r_0\ dt = -r_0 t \tag{44}$$

og

$$g(t) = -u_0 - \alpha t \tag{45}$$

Vi har altså, at

$$\int \exp(F(t))g(t)\ dt = \tag{46}$$

$$\int \exp(-r_0 t)(-u_0 - \alpha t)\ dt = \tag{47}$$

$$\left(-u_0 \int \exp(-r_0 t)\ dt\right) + \left(-\alpha \int \exp(-r_0 t)t\ dt\right) = \tag{48}$$

$$\left(-u_0 \frac{-1}{r_0} \exp(-r_0 t)\right) + \left(-\alpha \frac{-\exp(-r_0 t)(r_0 t + 1)}{r_0^2}\right) + C = \tag{49}$$

$$\left(\frac{r_0 u_0 + \alpha}{r_0^2} + \frac{\alpha}{r_0} t\right) \exp(-r_0 t) + C = \tag{50}$$

$$h(t) \exp(-r_0 t) + C \tag{51}$$

hvor

$$h(t) = \frac{r_0 u_0 + \alpha}{r_0^2} + \frac{\alpha}{r_0} t \tag{52}$$

og

$$C \in \mathbb{R} \tag{53}$$

Altså har vi alt i alt nu, at den fuldstændige løsning er givet ved

$$x(t) = \tag{54}$$

$$\exp(-F(t)) \int \exp(F(t))g(t)\ dt = \tag{55}$$

$$\exp(r_0 t)\Big(h(t) \exp(-r_0 t) + C\Big) = \tag{56}$$

$$h(t) + C \exp(r_0 t) \tag{57}$$

3.1.2 (ii)

Vi har, at

$$x(0) = h(0) + C\exp(r_0 \cdot 0) = h(0) + C \tag{58}$$

hvilket giver, at

$$C = x(0) - h(0) \tag{59}$$

Altså kan vi også skrive den fuldstændige løsning som

$$x(t) = \tag{60}$$
$$h(t) + (x(0) - h(0))\exp(r_0 t) = \tag{61}$$
$$h(t) + \left(x(0) - \frac{r_0 u_0 + \alpha}{r_0^2}\right)\exp(r_0 t) \tag{62}$$

3.1.3 (iii)

Vi ser, at

$$h(t) = \frac{r_0 u_0 + \alpha}{r_0^2} + \frac{\alpha}{r_0} t \tag{63}$$

er en linær funktion af t. Desuden er den voksende, da α og r_0 begge er skarpt positive. Altså går h mod uendelig, når t går mod uendelig. Hvis $C = 0$, så går

$$x(t) = h(t) + C\exp(r_0 t) \tag{64}$$

altså mod uendelig, når t går mod uendelig. Vi ved desuden, at hvis $C \neq 0$, så

$$\frac{h(t)}{C\exp(r_0 t)} \to 0, \qquad t \to \infty \tag{65}$$

da en eksponentiel funtion med positiv rate dominerer enhver linær funktion i det lange løb. Hvis $C \neq 0$, så har vi altså, at $x(t)$ går i minus på lang sigt, hvis og kun hvis $C < 0$. Altså har vi alt i alt, at $x(t)$ undgår at gå i minus på lang sigt, hvis og kun hvis $0 \leq C$, hvilket gælder, hvis og kun hvis

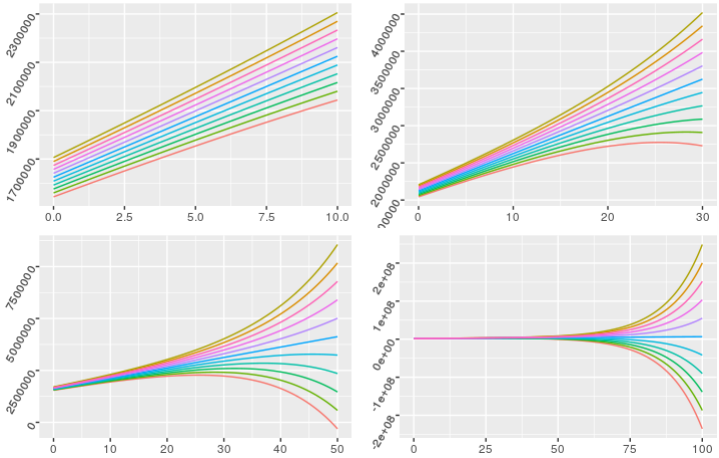
$$h(0) \leq x(0) \tag{66}$$

Altså har vi, at

$$x_{\min} = h(0) = \frac{r_0 u_0 + \alpha}{r_0^2} \tag{67}$$

3.1.4 (iv)

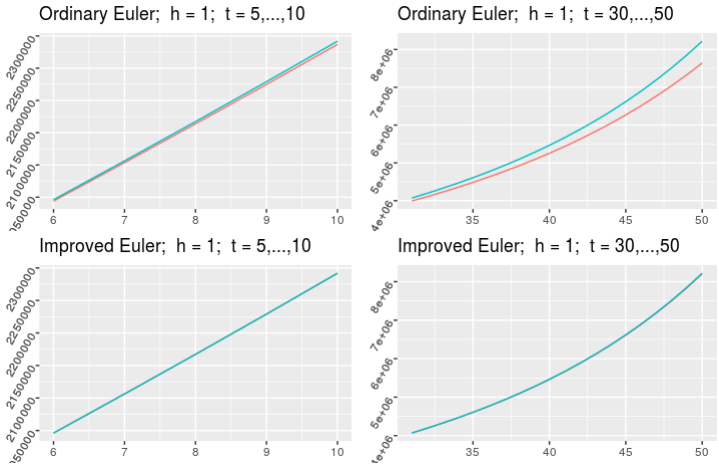
Her er mine plots:



Hvert plot viser $x(t)$ for $x_0 = 0.95x_{\min}, 0.96x_{\min}, ..., x_{\min}, ..., 1.04x_{\min}, 1.05x_{\min}$. Plottene adskiller sig ved at vise funktionernes udvikling for højere og højere t. Vi ser, at hvor alle funktionerne ser ret linære ud og følges nogenlunde ad for $t = 0, ..., 10$, så bliver de mere og mere eksponentielle og adskiller de sig mere og mere fra hinanden jo større t bliver.

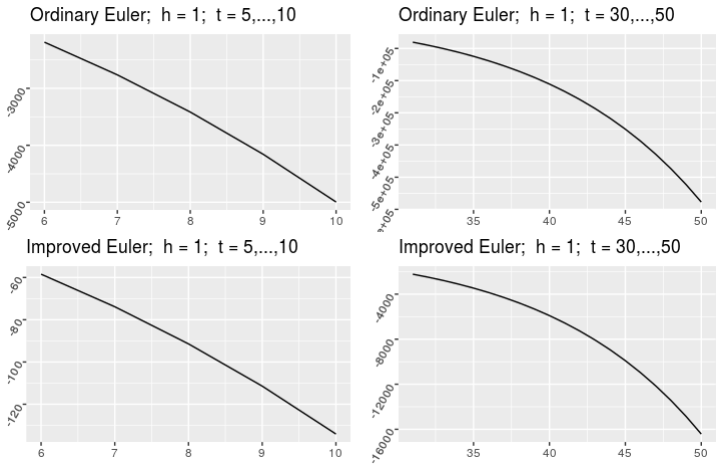
3.1.5 (iv & v)

Her er nogle plots af numeriske approksimationer af $x(t)$ ved hjælp af Eulers og Eulers forbedrede metode sammenlignet med den sande $x(t)$. I hvert plot er den blå linje den sande $x(t)$ og den røde linje er den numeriske approksimation:

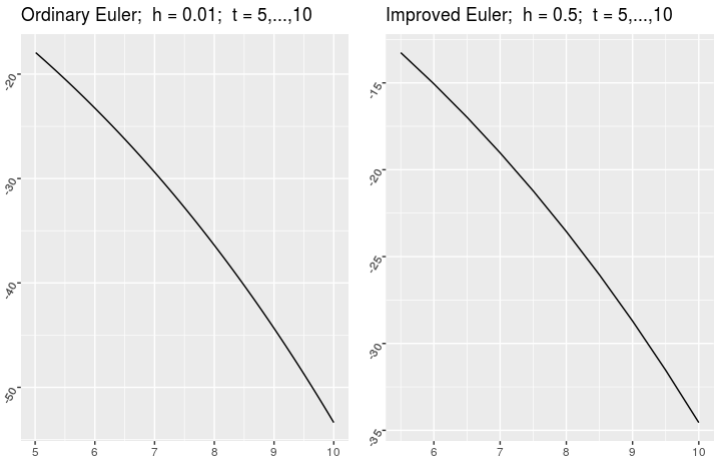


I alle plottene er $h = 1$, men vi ser på forskellige intervaller af t. Generelt ser vi, at Eulers forberede metode virkelig ser ud til at præstere meget bedre - faktisk i så høj grad, at grafen for den sande $x(t)$ ender med at ligge ovenpå grafen for den numeriske approksimation. Vi ser også, at de numeriske approksimationer underestimeterer den sande funktion, hvilket er forventeligt, da den sande funktion er konveks.

Her er nogle plots af, hvor store fejl de numeriske approksimationer hver især begår (fejlen er udregnet som approksimerede værdi af $x(t)$ minus den sande værdi af $x(t)$):



Fejlene er negative, da approksimationerne som sagt underestimeterer den sande værdi. Vi ser, at for $t = 10$ har Eulers normale metode en fejl på omkring -5000, mens den forbedrede metode har en fejl på omkring -130. Her er nogle plots, der viser, at hvis vi skal begrænse den absolutte værdi af fejlen for $t = 10$ til 100, så er det i hvert fald nok at sætte $h = 0.01$ for den normale og $h = 0.5$ for den forbedrede metode:



Ved hjælp af uniroot bestemmer jeg, at for $h = 0.0188$, så er fejlen -100 for $t = 10$ for den normale metode. Ligeledes bestemmer jeg, at for $h = 0.847$, så er fejlen -100 for $t = 10$ for den forberedrede metode.

3.2 Delopgave b

3.2.1 (i)

Vi har den inhomogene, linære 1. ordens differentialligning

$$x'(t) = r_0 x(t) - u_0 \exp(\beta t) \tag{68}$$

som vi kan omskrive til

$$\frac{dx}{dt} + (-r_0 x) = -u_0 \exp(\beta t) \tag{69}$$

hvorved den er på samme form, som kursusbogen Matematik for Biovidenskab opskriver inhomogene, linære 1. ordens differentialligninger. Hermed kan vi, som i delopgave a, løse den med panserformlen. Jeg springer mellemregningerne over og giver den fuldstændige løsning, som vi hermed kommer frem til:

$$x(t) = j(t) + C \exp(r_0 t) \tag{70}$$

hvor

$$j(t) = \frac{u_0}{r - \beta} \exp(\beta t) \tag{71}$$

og

$$C \in \mathbb{R} \tag{72}$$

3.2.2 (ii)

Vi har, at

$$x(0) = j(0) + C\exp(r_0 \cdot 0) = j(0) + C \tag{73}$$

hvilket giver, at

$$C = x(0) - j(0) \tag{74}$$

Altså kan vi også skrive den fuldstændige løsning som

$$x(t) = \tag{75}$$
$$j(t) + (x(0) - j(0))\exp(r_0 t) = \tag{76}$$
$$j(t) + \left(x(0) - \frac{u_0}{r_0 - \beta}\right)\exp(r_0 t) \tag{77}$$

3.2.3 (iii)

Vi ser, at

$$j(t) = \frac{u_0}{r - \beta}\exp(\beta t) \tag{78}$$

er en eksponentiel funktion af t. Altså er $x(t)$ en sum af to eksponentielle funktioner. Det gælder generelt om to eksponentielle funktioner $a\exp(\gamma t)$ pg $b\exp(\kappa t)$, at hvis $\gamma > \kappa$, så

$$\frac{a\exp(\gamma t)}{b\exp(\kappa t)} \to 0, \qquad t \to \infty \tag{79}$$

Hvis vi antager, at $\beta > r$, har vi altså, at $j(t)$ dominerer udtrykket for $x(t)$, når t går mod uendelig, og $j(t)$ er en eksponentiel funktion med negativt fortegn. Altså vil $j(t)$ og dermed $x(t)$ gå mod minus uendelig, når t går mod uendelig.

Hvis vi derimod antager, at $r > \beta$, så går

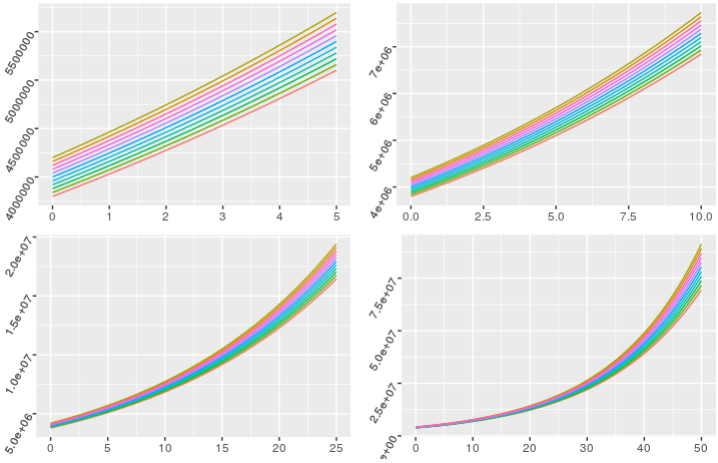
$$x(t) = j(t) + (x(0) - j(0))\exp(r_0 t) \tag{80}$$

mod uendelig, hvis $j(0) \leq x(0)$, og $x(t)$ går mod minus uendelig, hvis $x(0) < j(0)$. Altså har vi, at

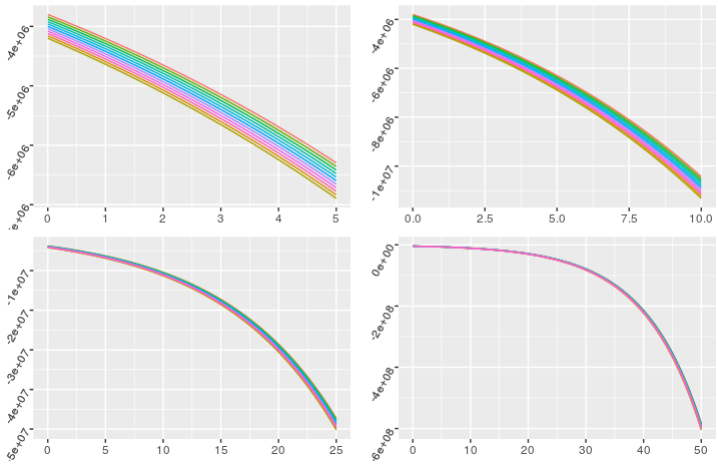
$$x_{\min} = j(0) = \frac{u_0}{r_0 - \beta} \tag{81}$$

3.2.4 (iv)

Her er mine plots for delspørgsmål (A):



Her er mine plots for delspørgsmål (B):



```
knitr::opts_knit$set(root.dir = '~/Desktop/bio_modelling_course/mod3/proj3/')
```

Code for question 2

```
library(magrittr)
library(tidyverse)
```

```
## ── Attaching packages ─────────────────────────────────────────────────────── tidyverse 1.2.1 ──
```

```
## ✔ ggplot2 2.2.1     ✔ purrr   0.2.4
## ✔ tibble  1.4.2     ✔ dplyr   0.7.4
## ✔ tidyr   0.8.0     ✔ stringr 1.3.0
## ✔ readr   1.1.1     ✔ forcats 0.3.0
```

```
## ── Conflicts ────────────────────────────────────────────────────────── tidyverse_conflicts() ──
## ✖ tidyr::extract()   masks magrittr::extract()
## ✖ dplyr::filter()    masks stats::filter()
## ✖ dplyr::lag()       masks stats::lag()
## ✖ purrr::set_names() masks magrittr::set_names()
```

```
library(gridExtra)
```

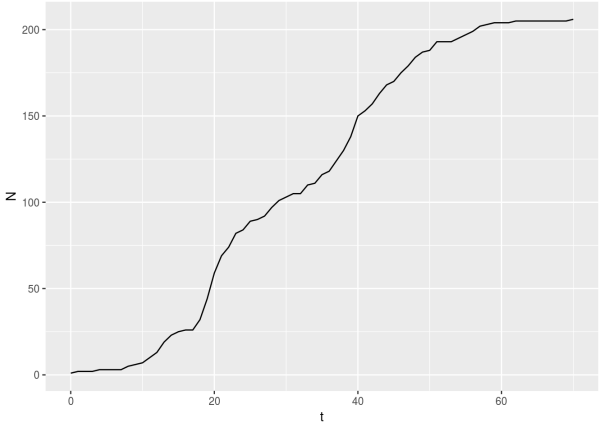
```
##
## Attaching package: 'gridExtra'
```

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

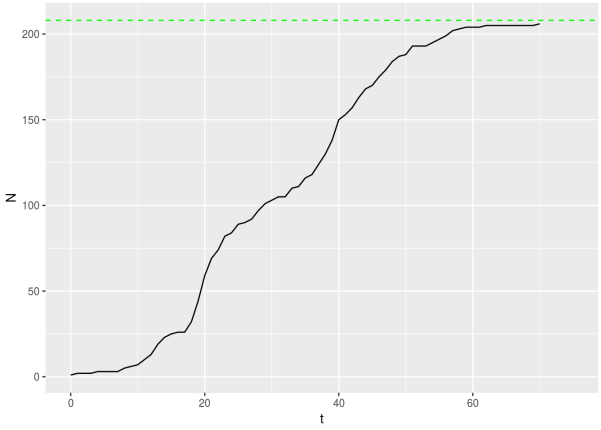
```
sars<-read_table('SARS.txt') %>%
  mutate_all(funs(as.integer))
```

```
## Parsed with column specification:
## cols(
##   Dag = col_integer(),
##   `Smittede  ` = col_integer()
## )
```

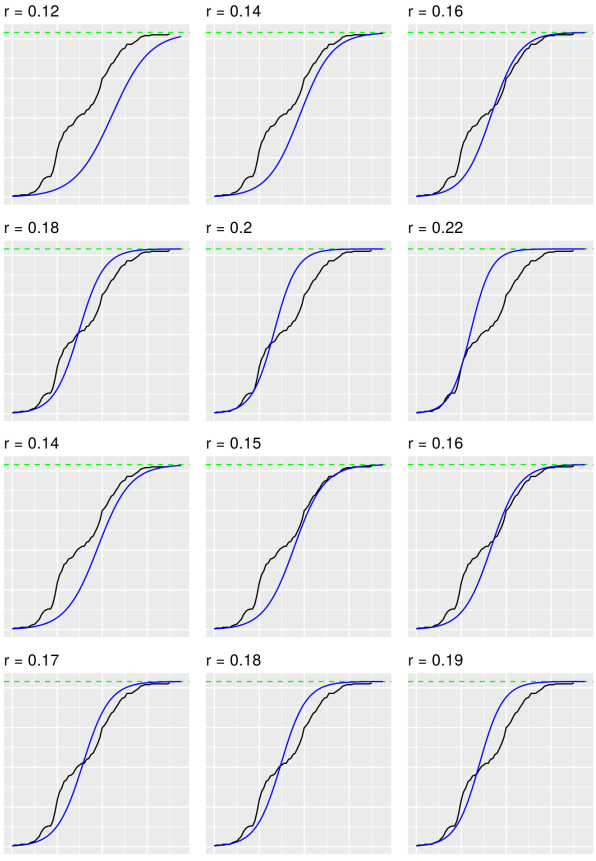
```
colnames(sars) <- c('t', 'N')

plt <- sars %>%
  ggplot(aes(t, N)) +
  geom_line()
plt
```




```
plt2 <- plt +
  geom_hline(yintercept = 208,
             color='green',
             linetype='dashed') +
  xlim(0, 75)
plt2
```



```
make_logit <- function(r) {
  list(Vectorize(function(t) 208/(1 + 207*exp(-r*t))), r)
}

find_r_plots <- function(model_builder, search_spaces) {
  map(search_spaces,
      function (space) {
        grid.arrange(grobs=map(map(space, model_builder),
                               function(l) {
                                 plt2 +
                                   stat_function(fun=l[[1]], color='blue') +
                                   ggtitle(str_c('r = ', l[[2]])) +
                                   theme(axis.text = element_blank(),
                                         axis.ticks = element_blank(),
                                         axis.title = element_blank())
                               }),
                     nrow=2)
      }
  )
}

find_r_plots(make_logit, list(seq(0.12,0.22,by=0.02), seq(0.14,0.19,by=0.01)))
```

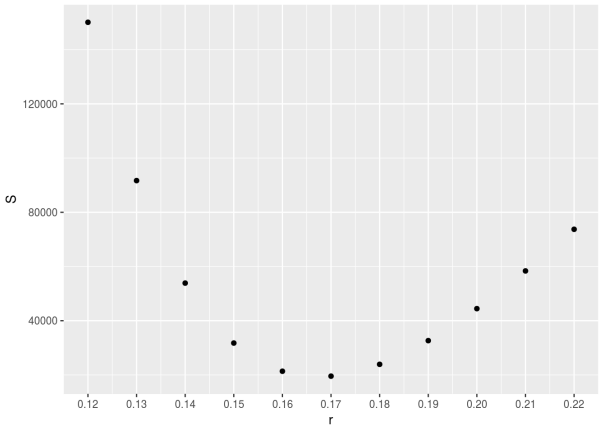



```
## [[1]]
## TableGrob (2 x 3) "arrange": 6 grobs
##   z     cells    name           grob
## 1 1 (1-1,1-1) arrange gtable[layout]
## 2 2 (1-1,2-2) arrange gtable[layout]
## 3 3 (1-1,3-3) arrange gtable[layout]
## 4 4 (2-2,1-1) arrange gtable[layout]
## 5 5 (2-2,2-2) arrange gtable[layout]
## 6 6 (2-2,3-3) arrange gtable[layout]
##
## [[2]]
## TableGrob (2 x 3) "arrange": 6 grobs
##   z     cells    name           grob
## 1 1 (1-1,1-1) arrange gtable[layout]
## 2 2 (1-1,2-2) arrange gtable[layout]
## 3 3 (1-1,3-3) arrange gtable[layout]
## 4 4 (2-2,1-1) arrange gtable[layout]
## 5 5 (2-2,2-2) arrange gtable[layout]
## 6 6 (2-2,3-3) arrange gtable[layout]
```

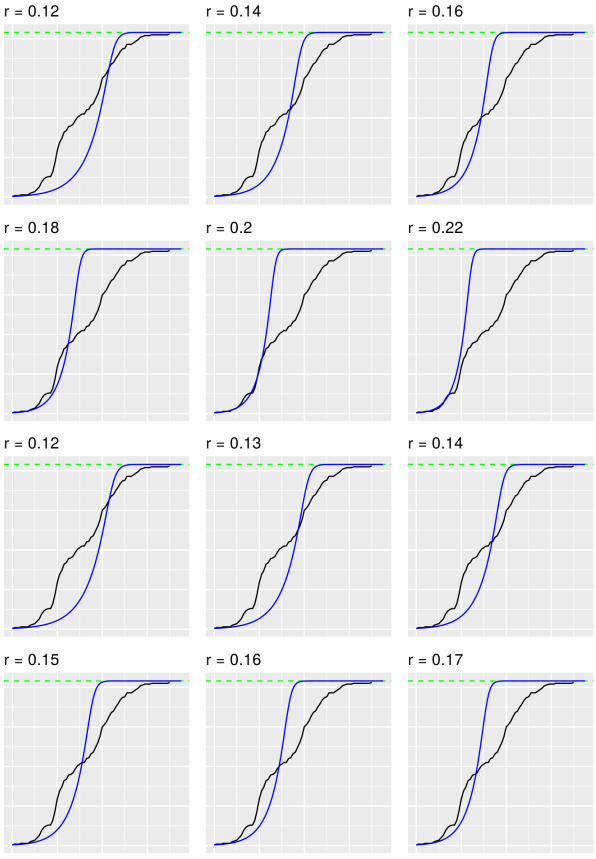
```
S <- function(model) {
  sars %>%
    mutate(pred = model(t),
           res_sqrd = (N-pred)**2) %>%
    summarise(S = sum(res_sqrd)) %>%
    .$S
}

S(make_logit(0.17)[[1]])
```

```
## [1] 19549.88
```

```
eval_mods <- function(rs, model_builder) {
  data_frame(r=rs) %>%
    mutate(S = map_dbl(map(rs, function(r) model_builder(r)[[1]]), S)) %>%
    ggplot(aes(r,S)) +
    geom_point() +
    scale_x_continuous(breaks=rs)
}

eval_mods(seq(.12,.22,by=.01), make_logit)
```

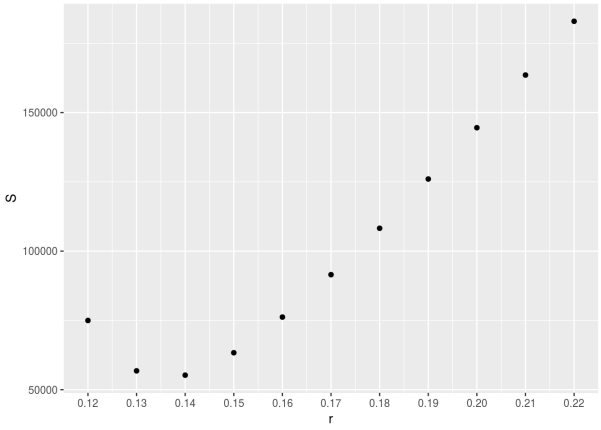



```
make_mod_logit <- function(r) {
  list(Vectorize(function(t) 208/((1 + (208**5 - 1)*exp(-5*r*t)))**(1/5)), r)
}

find_r_plots(make_mod_logit, list(seq(0.12,0.22,by=0.02), seq(0.12,0.17,by=0.01)))
```




```
## [[1]]
## TableGrob (2 x 3) "arrange": 6 grobs
##   z     cells    name           grob
## 1 1 (1-1,1-1) arrange gtable[layout]
## 2 2 (1-1,2-2) arrange gtable[layout]
## 3 3 (1-1,3-3) arrange gtable[layout]
## 4 4 (2-2,1-1) arrange gtable[layout]
## 5 5 (2-2,2-2) arrange gtable[layout]
## 6 6 (2-2,3-3) arrange gtable[layout]
##
## [[2]]
## TableGrob (2 x 3) "arrange": 6 grobs
##   z     cells    name           grob
## 1 1 (1-1,1-1) arrange gtable[layout]
## 2 2 (1-1,2-2) arrange gtable[layout]
## 3 3 (1-1,3-3) arrange gtable[layout]
## 4 4 (2-2,1-1) arrange gtable[layout]
## 5 5 (2-2,2-2) arrange gtable[layout]
## 6 6 (2-2,3-3) arrange gtable[layout]
```

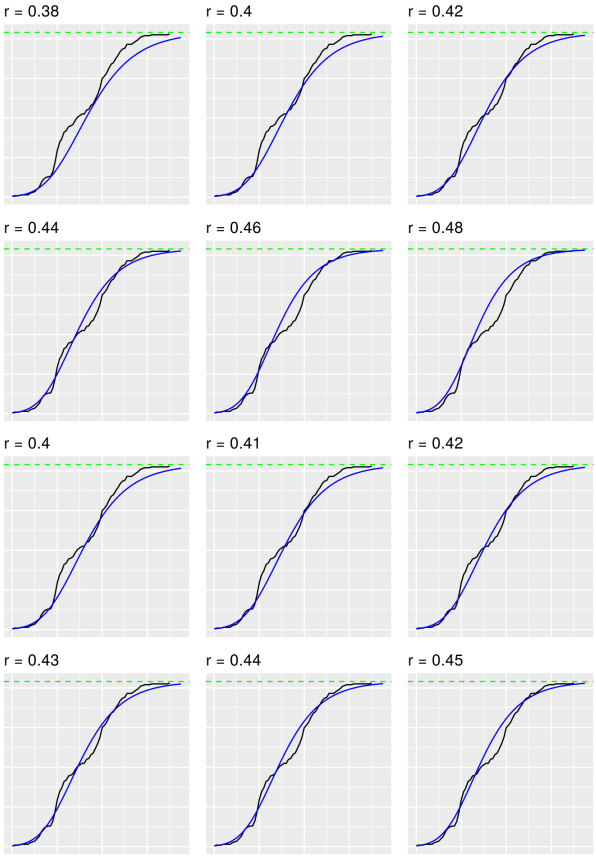
```
eval_mods(seq(.12,.22,by=.01), make_mod_logit)
```




```
S(make_mod_logit(0.14)[[1]])
```

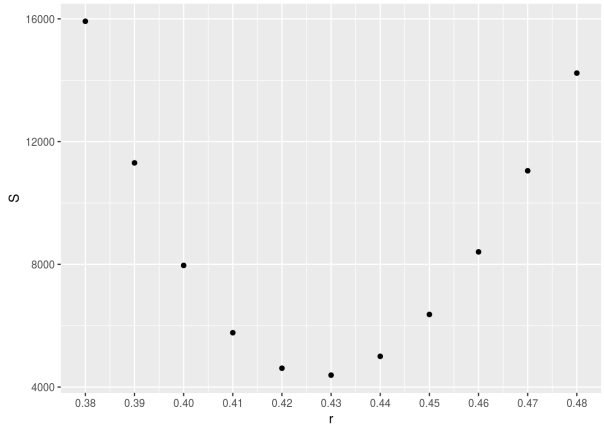
```
## [1] 55249.06
```

```
make_mod_logit_2 <- function(r) {
  list(Vectorize(function(t) 208/((1 + 1.908*exp(-0.2*r*t)))**5), r)
}

find_r_plots(make_mod_logit_2, list(seq(0.38,0.48,by=0.02), seq(0.40,0.45,by=0.01)))
```




```
## [[1]]
## TableGrob (2 x 3) "arrange": 6 grobs
##   z     cells    name           grob
## 1 1 (1-1,1-1) arrange gtable[layout]
## 2 2 (1-1,2-2) arrange gtable[layout]
## 3 3 (1-1,3-3) arrange gtable[layout]
## 4 4 (2-2,1-1) arrange gtable[layout]
## 5 5 (2-2,2-2) arrange gtable[layout]
## 6 6 (2-2,3-3) arrange gtable[layout]
##
## [[2]]
## TableGrob (2 x 3) "arrange": 6 grobs
##   z     cells    name           grob
## 1 1 (1-1,1-1) arrange gtable[layout]
## 2 2 (1-1,2-2) arrange gtable[layout]
## 3 3 (1-1,3-3) arrange gtable[layout]
## 4 4 (2-2,1-1) arrange gtable[layout]
## 5 5 (2-2,2-2) arrange gtable[layout]
## 6 6 (2-2,3-3) arrange gtable[layout]
```

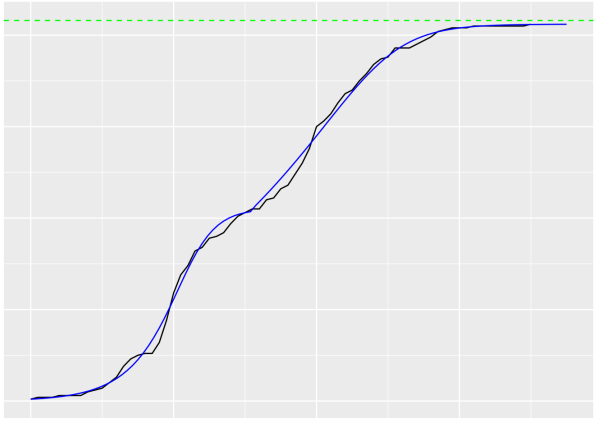
```
eval_mods(seq(0.38,0.48,by=0.01), make_mod_logit_2)
```




```
S(make_mod_logit_2(0.43)[[1]])
```

```
## [1] 4388.52
```

```
N1 <- function(t) 105/((1 + 10182*exp(-0.415*t)))**(0.504)
N2 <- function(t) 206/((1 + 146757*exp(-0.243*t)))**(0.154)

last_mod <- Vectorize(function(t) {
  if (t <= 31) N1(t)
  else N2(t)
})

plt2 +
  stat_function(fun=last_mod, color='blue') +
  theme(axis.text = element_blank(),
        axis.ticks = element_blank(),
        axis.title = element_blank())
```



```
c(N1(31), N2(31))
```

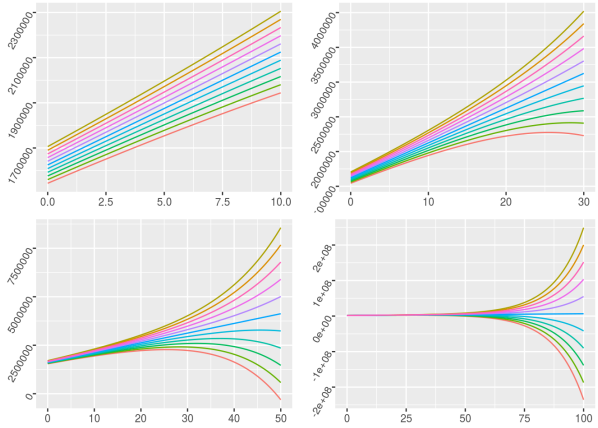
```
## [1] 103.6330 104.9982
```

```
S(last_mod)
```

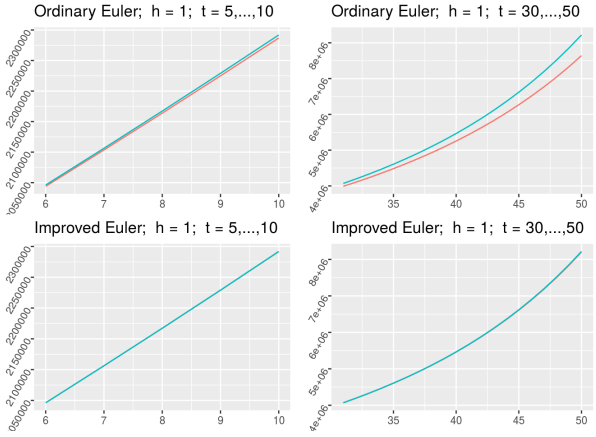
```
## [1] 667.2926
```

Code for question 3

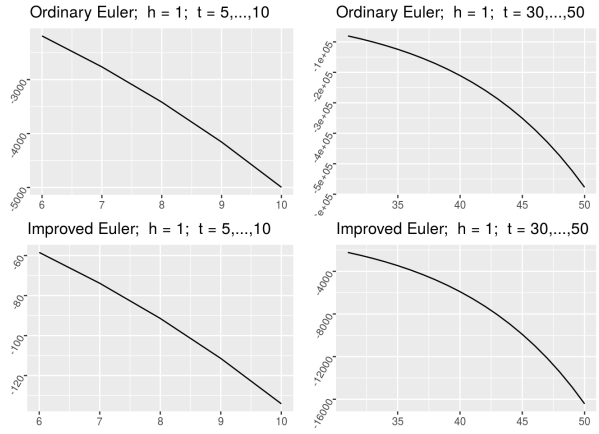
```
library(magrittr)
library(tidyverse)
library(gridExtra)

r0 <- 0.08
u0 <- 80000
alpha <- 4000

xmin <- (r0*u0 + alpha)/r0**2

h <- Vectorize(function(t) {
  xmin + (alpha/r0)*t
})

x <- function(t, x0) {
  h(t) + (x0 - h(0))*exp(r0*t)
}
ts <- seq(0,100,by=0.2)
x0s <- xmin*seq(0.95,1.05,by=0.01)
xs <- map(map(x0s, function(x0) {function(t) x(t, x0)}),
    function(f) {f(ts)})
xs <- data.frame(xs)
colnames(xs) <- str_c('x', as.character(1:11))
plot_lines <- function(max_t) {
  xs %>%
    mutate(t = ts) %>%
    filter(t <= max_t) %>%
    gather(key='key', value='value', -t) %>%
    ggplot(aes(t, value, color=key)) +
    geom_line() +
    theme(legend.position = 'none',
          axis.text.y = element_text(angle = 60),
          axis.title = element_blank())
}
grid.arrange(grobs=map(c(10,30,50,100), plot_lines), nrow=2)
```



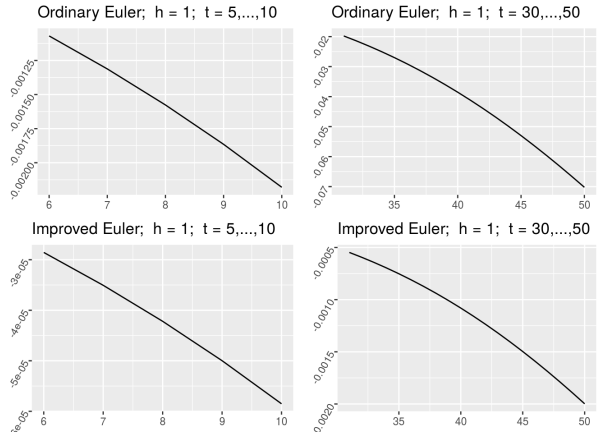
```
#########################################################################

euler <- function(diff,
                  x0, y0, x_end,
                  h, improv = F) {
  xs <- seq(x0, x_end + x_end%%h, by = h)
  ys <- rep(NA, length(xs))
  ys[1] <- y0
  for (i in 2:length(xs)) {
    ystar <- ys[i-1] + diff(xs[i-1], ys[i-1])*h
    if (improv) {
      ys[i] <- ys[i-1] + (diff(xs[i-1], ys[i-1]) + diff(xs[i], ystar))*h/2
    }
    else {
      ys[i] <- ystar
    }
  }
  data_frame(t=xs, x_euler=ys)
}

approx <- function(h, t_end = 10, improv = F) {
  euler(function(t, x) r0*x - u0 - alpha*t,
        0, 1700000, t_end,
        h, improv) %>%
    mutate(x_true = map_dbl(t, function(t) x(t,1700000)),
           error = x_euler - x_true,
           error_percent = error / x_true)
}

approx_plot <- function(h, t_end, improv, min_t) {
  if (improv) {
    name <- 'Improved Euler'
  }
  else {
    name <- 'Ordinary Euler'
  }
  approx(h, t_end, improv) %>%
    filter(min_t<t) %>%
    select(-error, -error_percent) %>%
    gather(key='key', value='value', -t) %>%
    ggplot(aes(t, value, color=key)) +
    geom_line() +
    theme(legend.position = 'none',
          axis.text.y = element_text(angle=60),
          axis.title = element_blank()) +
    ggtitle(str_c(name, ';  h = ', h, ';  t = ', min_t, ',...,', t_end))
}

error_plot <- function(h, t_end, improv, min_t, error_type) {
  if (improv) {
    name <- 'Improved Euler'
  }
  else {
    name <- 'Ordinary Euler'
  }
  approx(h, t_end, improv) %>%
    filter(min_t<t) %>%
    ggplot(aes_string('t', error_type)) +
    geom_line() +
    theme(axis.text.y = element_text(angle=60),
          axis.title = element_blank()) +
    ggtitle(str_c(name, ';  h = ', h, ';  t = ', min_t, ',...,', t_end))
}

grid.arrange(grobs = list(approx_plot(1, 10, F, 5),
                          approx_plot(1, 50, F, 30),
                          approx_plot(1, 10, T, 5),
                          approx_plot(1, 50, T, 30)),
             nrow=2)
```



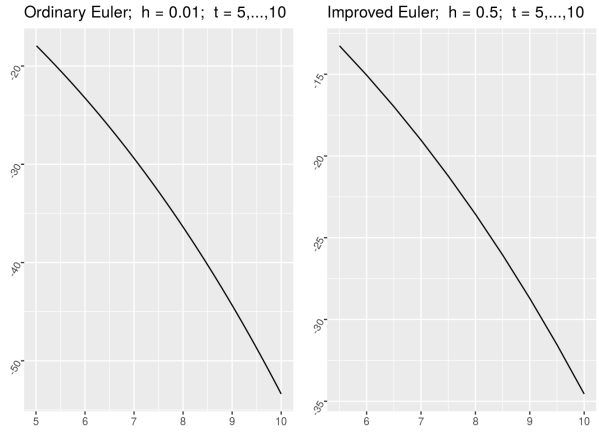

```
grid.arrange(grobs = list(error_plot(1, 10, F, 5, 'error'),
                          error_plot(1, 50, F, 30, 'error'),
                          error_plot(1, 10, T, 5, 'error'),
                          error_plot(1, 50, T, 30, 'error')),
             nrow=2)
```




```
grid.arrange(grobs = list(error_plot(1, 10, F, 5, 'error_percent'),
                          error_plot(1, 50, F, 30, 'error_percent'),
                          error_plot(1, 10, T, 5, 'error_percent'),
                          error_plot(1, 50, T, 30, 'error_percent')),
             nrow=2)
```




```
grid.arrange(grobs = list(error_plot(0.01, 10, F, 5, 'error'),
                          error_plot(0.5, 10, T, 5, 'error')),
             nrow=1)
```




```
h_threshold_ord <- uniroot(function(h) approx(h) %>% .$error %>% tail(1) + 100, c(0.01, 1))$root
h_threshold_improv <- uniroot(function(h) approx(h,improv=T) %>% .$error %>% tail(1) + 100, c(0.5, 1))$root
h_threshold_ord
```

```
## [1] 0.01875349
```

```
h_threshold_improv
```

```
## [1] 0.8469925
```

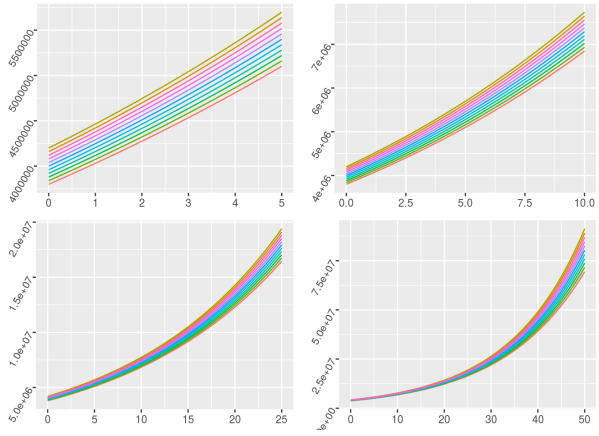
```
approx(h_threshold_ord, improv=F) %>% tail(1)
```

```
## # A tibble: 1 x 5
##       t  x_euler   x_true  error error_percent
##   <dbl>    <dbl>    <dbl>  <dbl>         <dbl>
## 1 10.00 2291538. 2291638. -100.0    -0.0000436
```

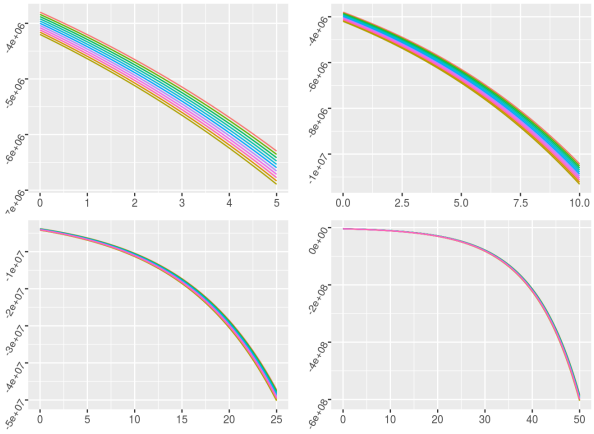
```
approx(h_threshold_improv, improv=T) %>% tail(1)
```

```
## # A tibble: 1 x 5
##       t  x_euler   x_true  error error_percent
##   <dbl>    <dbl>    <dbl>  <dbl>         <dbl>
## 1  10.2 2302214. 2302314. -100.0    -0.0000434
```

```
################################################################################

r0 <- 0.08
u0 <- 80000
beta <- 0.06
xmin <- u0/(r0-beta)

h <- Vectorize(function(t) {
  xmin*exp(beta*t)
})

x <- function(t, x0) {
  h(t) + (x0 - h(0))*exp(r0*t)
}
ts <- seq(0,500,by=0.2)
x0s <- xmin*seq(0.95,1.05,by=0.01)
xs <- map(map(x0s, function(x0) {function(t) x(t, x0)}),
          function(f) {f(ts)})
xs <- data.frame(xs)
colnames(xs) <- str_c('x', as.character(1:11))
plot_lines <- function(tlims) {
  xs %>%
    mutate(t = ts) %>%
    filter(tlims[1] <= t , t <= tlims[2]) %>%
    gather(key='key', value='value', -t) %>%
    ggplot(aes(t, value, color=key)) +
    geom_line() +
    theme(legend.position = 'none',
          axis.text.y = element_text(angle = 60),
          axis.title = element_blank())
}
grid.arrange(grobs=map(list(c(0,5),c(0,10),c(0,25), c(0,50)), plot_lines), nrow=2)
```




```
r0 <- 0.08
u0 <- 80000
beta <- 0.1
xmin <- u0/(r0-beta)

h <- Vectorize(function(t) {
  xmin*exp(beta*t)
})

x <- function(t, x0) {
  h(t) + (x0 - h(0))*exp(r0*t)
}
ts <- seq(0,500,by=0.2)
x0s <- xmin*seq(0.95,1.05,by=0.01)
xs <- map(map(x0s, function(x0) {function(t) x(t, x0)}),
          function(f) {f(ts)})
xs <- data.frame(xs)
colnames(xs) <- str_c('x', as.character(1:11))
plot_lines <- function(tlims) {
  xs %>%
    mutate(t = ts) %>%
    filter(tlims[1] <= t , t <= tlims[2]) %>%
    gather(key='key', value='value', -t) %>%
    ggplot(aes(t, value, color=key)) +
    geom_line() +
    theme(legend.position = 'none',
          axis.text.y = element_text(angle = 60),
          axis.title = element_blank())
}
grid.arrange(grobs=map(list(c(0,5),c(0,10),c(0,25), c(0,50)), plot_lines), nrow=2)
```


-4e+06
-5e+06
-6e+06
-7e+06
0
1
2
3
4
5
-4e+06
-6e+06
-8e+06
-1e+07
0.0
2.5
5.0
7.5
10.0
-1e+07
-2e+07
-3e+07
-4e+07
-5e+07
0
5
10
15
20
25
0e+00
-2e+08
-4e+08
-6e+08
0
10
20
30
40
50