Day7 exercise solutions

Ali Movasati, Tristan Koning, Isabelle Caroline Rose Cretton

Oct. 31st, 2024

```
# Set global code chunk options
knitr::opts_chunk$set(warning = FALSE)
```

```
# load required libraries
library(skimr)
library(ggplot2)
library(ggpubr)
library(magrittr)
library(tidyr)
```

```
##
## Attaching package: 'tidyr'

## The following object is masked from 'package:magrittr':
##
##     extract
```

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(tibble)
library(lme4)
```

```
## Loading required package: Matrix

##
## Attaching package: 'Matrix'
```

```
## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack
```

```
library(lattice)
library(stringr)
library(sm)
```

```
## Package 'sm', version 2.2-6.0: type help(sm) for summary information
```

```
# define functions
`%notin%` <- Negate(`%in%`)
```

Problem 1

```
medflies <- read.table(file = "data/medflies.txt", sep = "\t", header = T)

medflies %<>% mutate_at(3,as.numeric)
head(medflies)
```

```
##   day  living mort.rate
## 1   0 1203646    0.0000
## 2   1 1203646    0.0014
## 3   2 1201913    0.0040
## 4   3 1197098    0.0051
## 5   4 1191020    0.0064
## 6   5 1183419    0.0075
```

1.A)

```
# generate descriptive tables
skim(medflies)
```

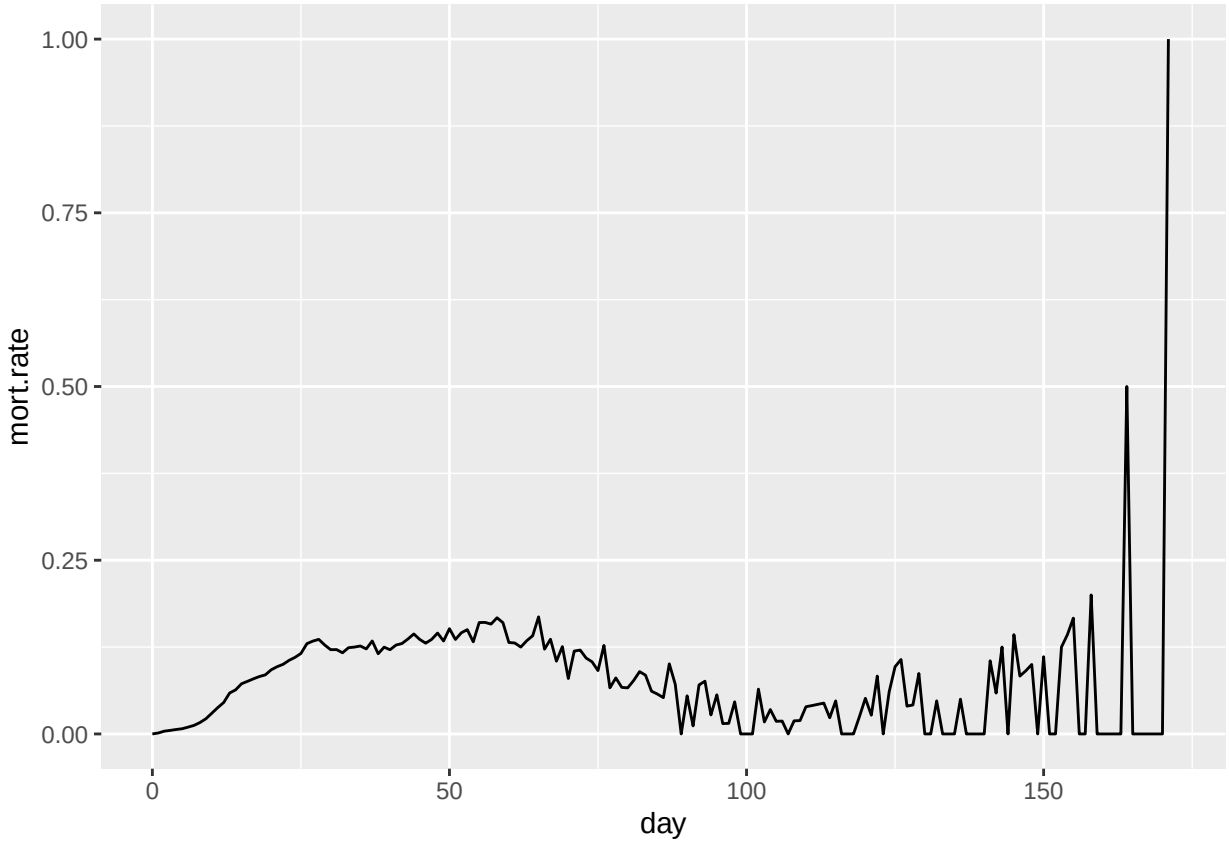
Table 1: Data summary

Name	medflies
Number of rows	173
Number of columns	3
Column type frequency:	
numeric	3
Group variables	None

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
day	0	1.00	86.00	50.08	0	43.00	86.00	129.00	172	
living	0	1.00	148501.07	339536.10	0	23.00	115.00	30360.00	1203646	
mort.rate	1	0.99	0.08	0.10	0	0.01	0.07	0.12	1	

```
# plot the two variables
medflies %>% ggplot(aes(x=day, y = mort.rate)) + geom_line()
```

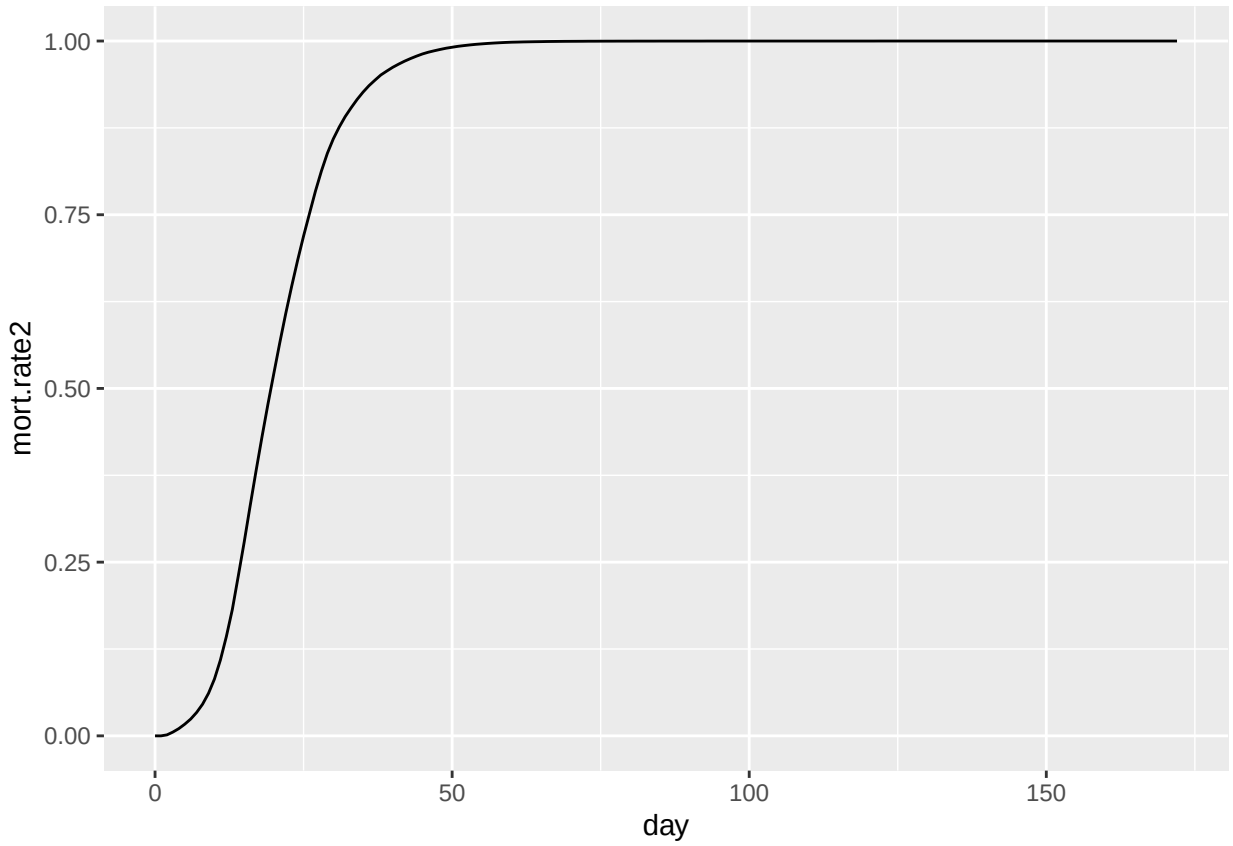


« comments »

The mortality data looks a little bit strange, there might be an error in calculations of the mortality rate!

1.B)

```
medflies %<>% mutate(mort.rate2 = (1203646-living)/1203646)


# plot the two variables
medflies %>% ggplot(aes(x=day, y = mort.rate2)) + geom_line()
```

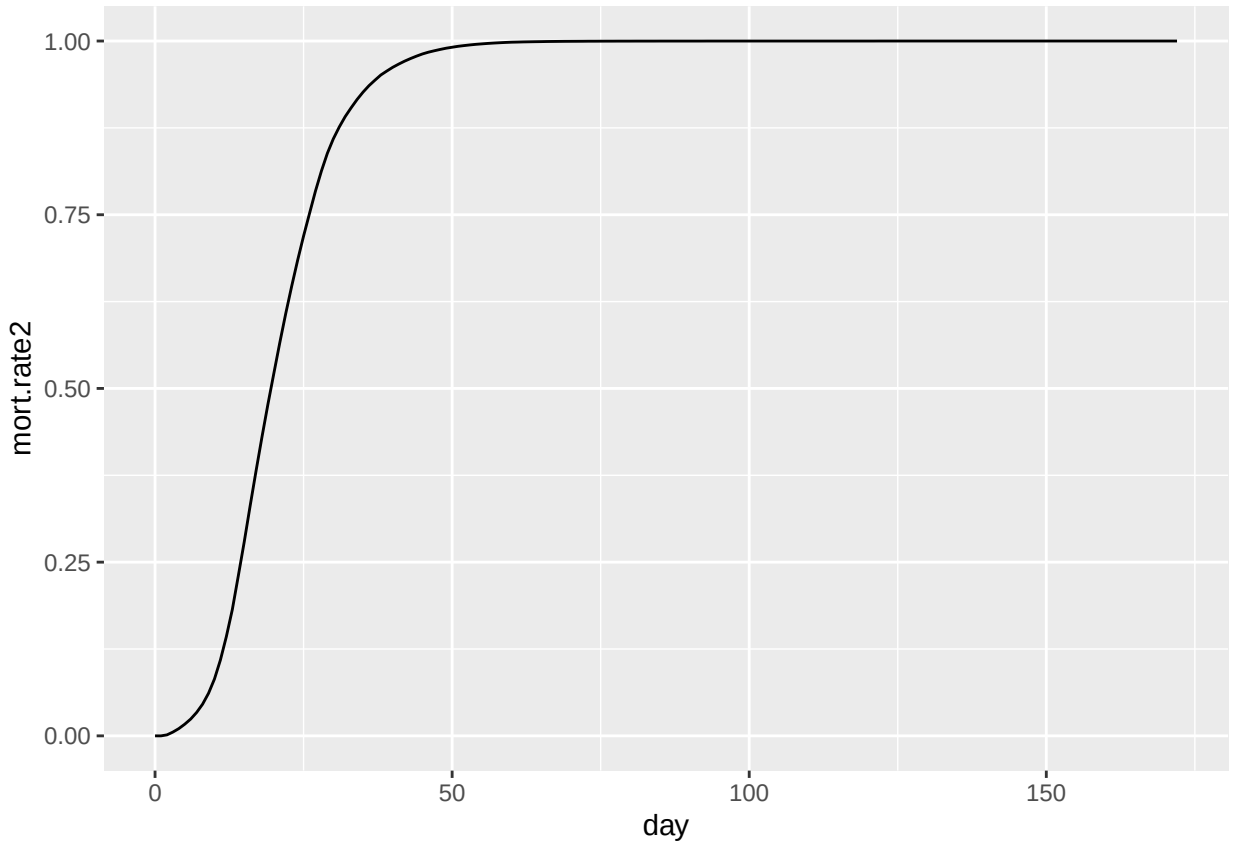


« comments »

No, the calculations were incorrect!

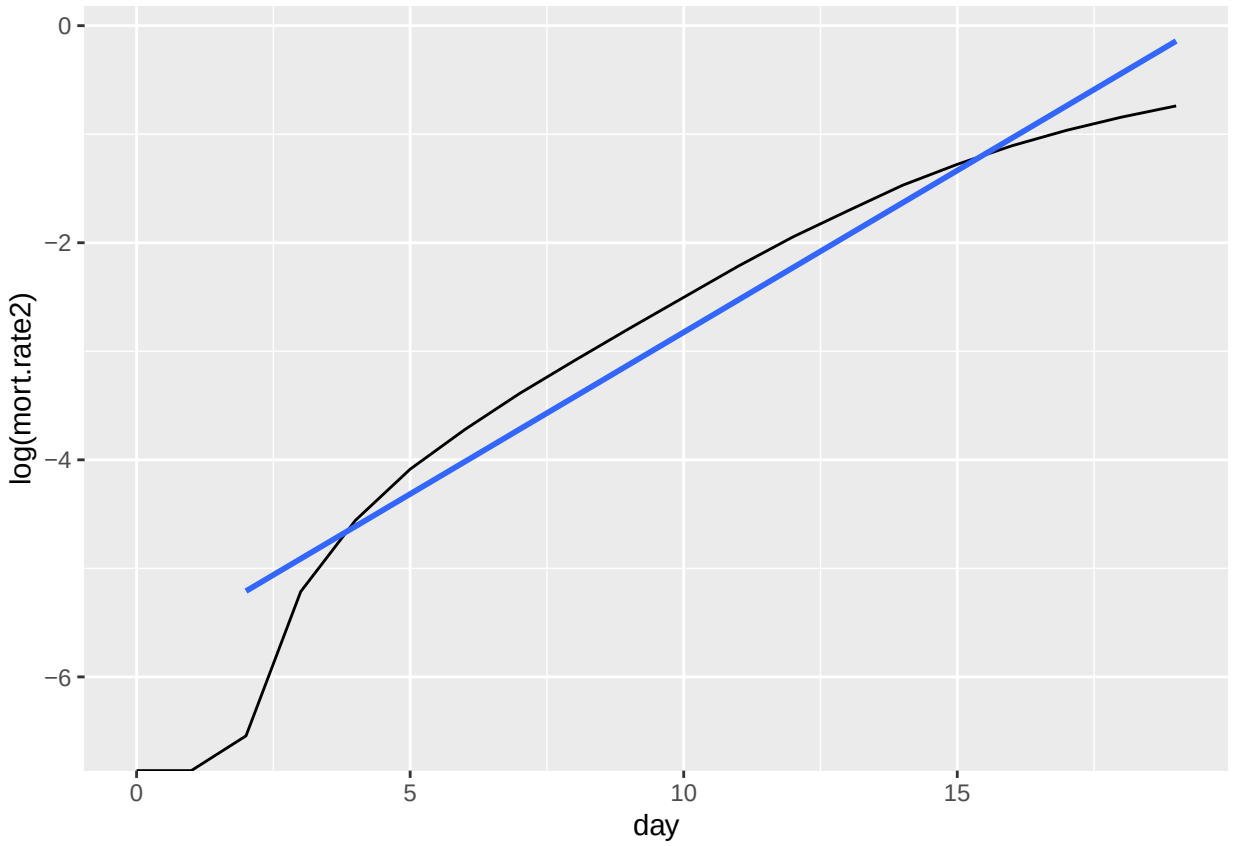
1.C)

```
# plot the two variables
medflies %>% ggplot(aes(x=day, y = mort.rate2)) + geom_line()
```

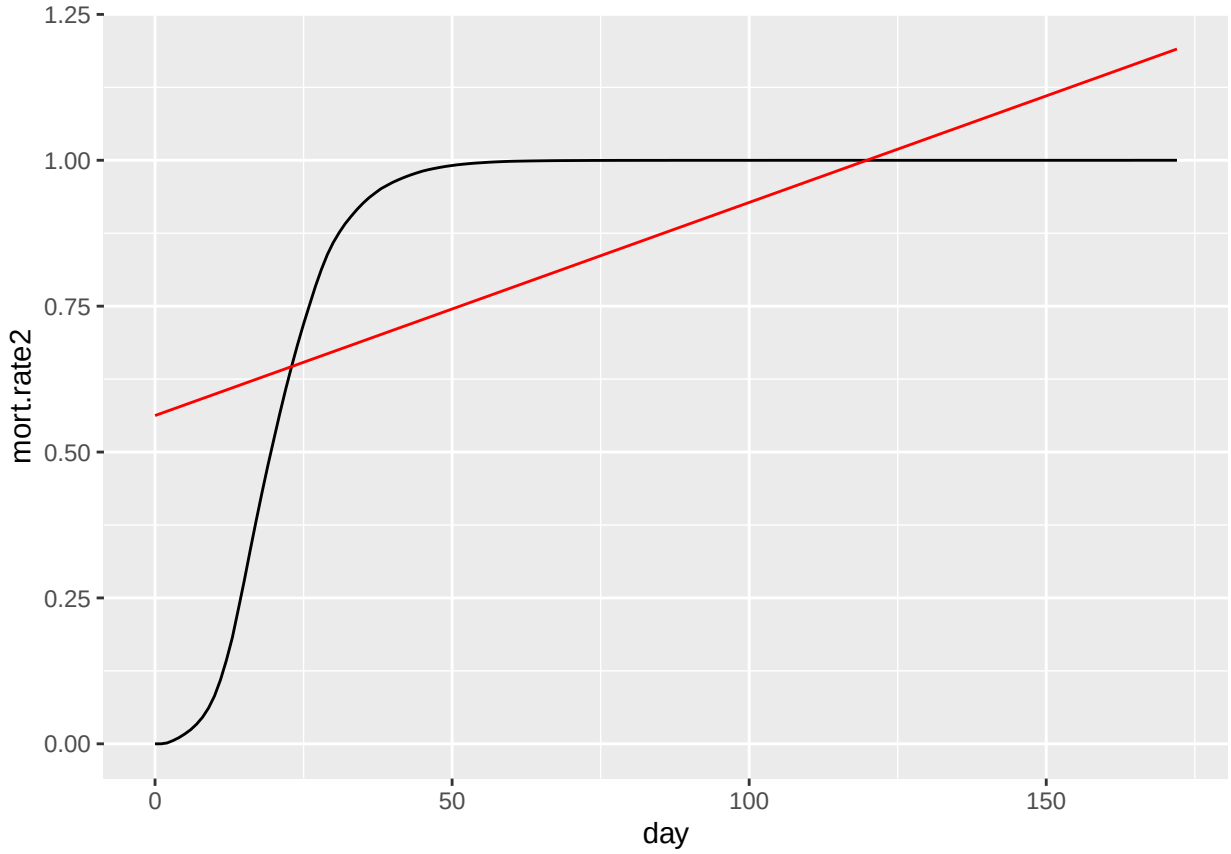



```
# first 20 days seem to be exponential
medflies[medflies$day < 20,] %>% ggplot(aes(x=day, y = log(mort.rate2))) + geom_line() + geom_smooth(me
```

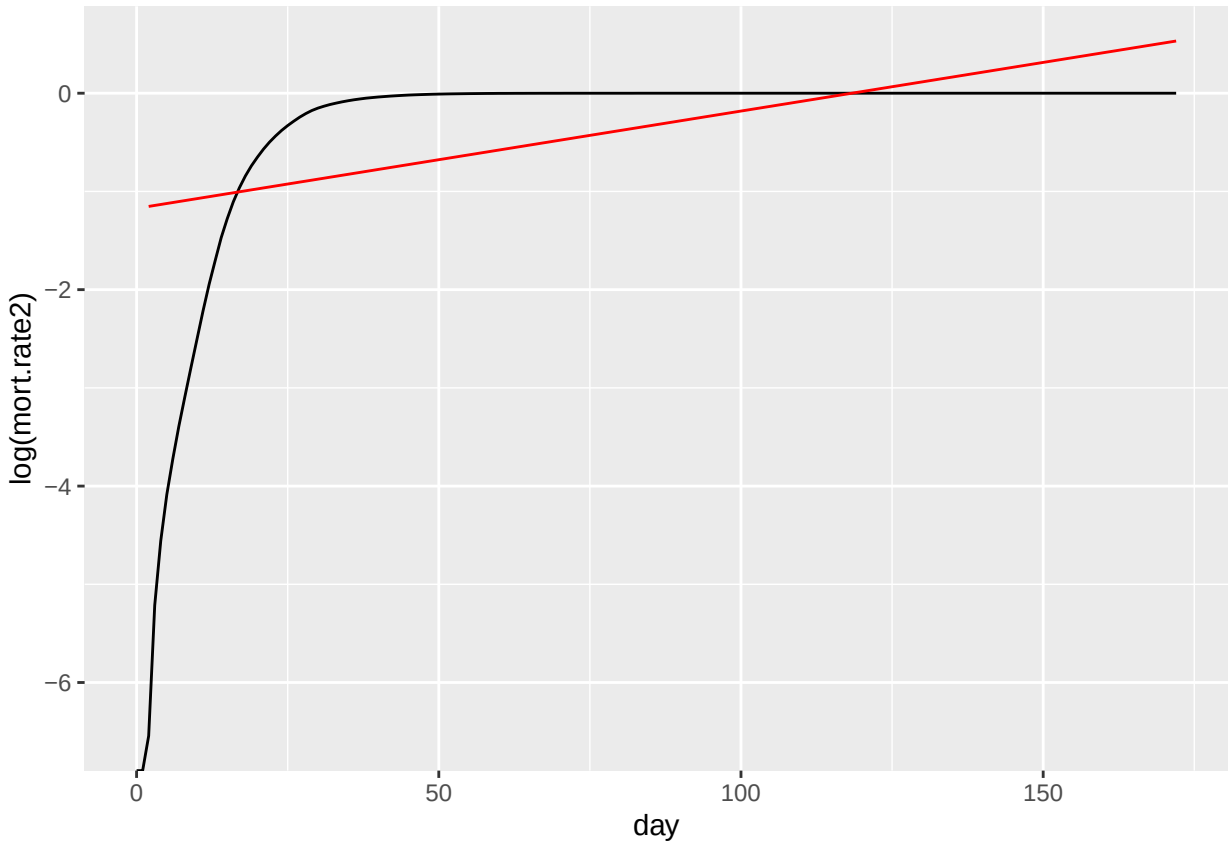
```
## `geom_smooth()` using formula = 'y ~ x'
```



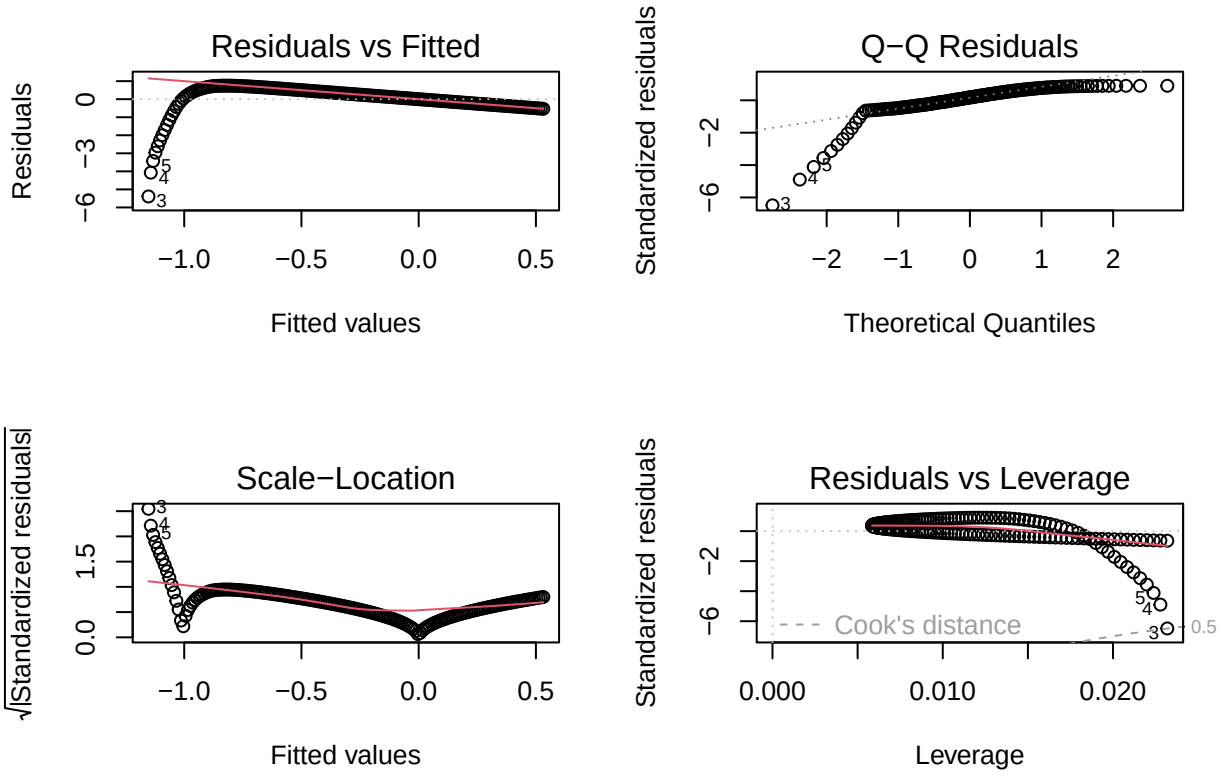

```
lm_model <- lm(mort.rate2~day, data=medflies)

medflies$predicted <- predict(lm_model, new_data = medflies)
medflies %>% ggplot(aes(x=day, y = mort.rate2)) + geom_line() + geom_line(aes(y = predicted), color = "
```




```
# model log(mort.rate)
lm_model_log <- lm(log(mort.rate2)~day, data=medflies[3:nrow(medflies),])

medflies$predicted_log <- c(NA,NA,predict(lm_model_log, new_data = medflies))
medflies %>% ggplot(aes(x=day, y = log(mort.rate2))) + geom_line() + geom_line(aes(y = predicted_log), c
```




```
par(mfrow = c(2, 2))  # Arrange 4 diagnostic plots
plot(lm_model_log)
```

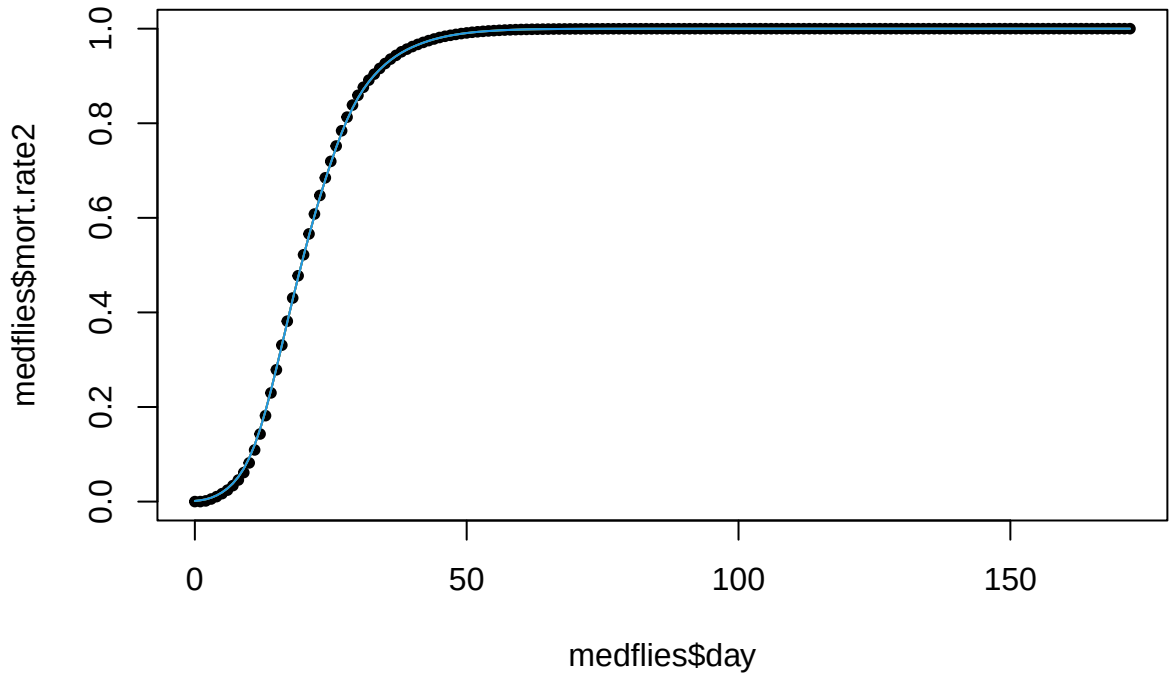


« comments »

The subset of data between days 3 to 25 seems to be exponential.

1.D)

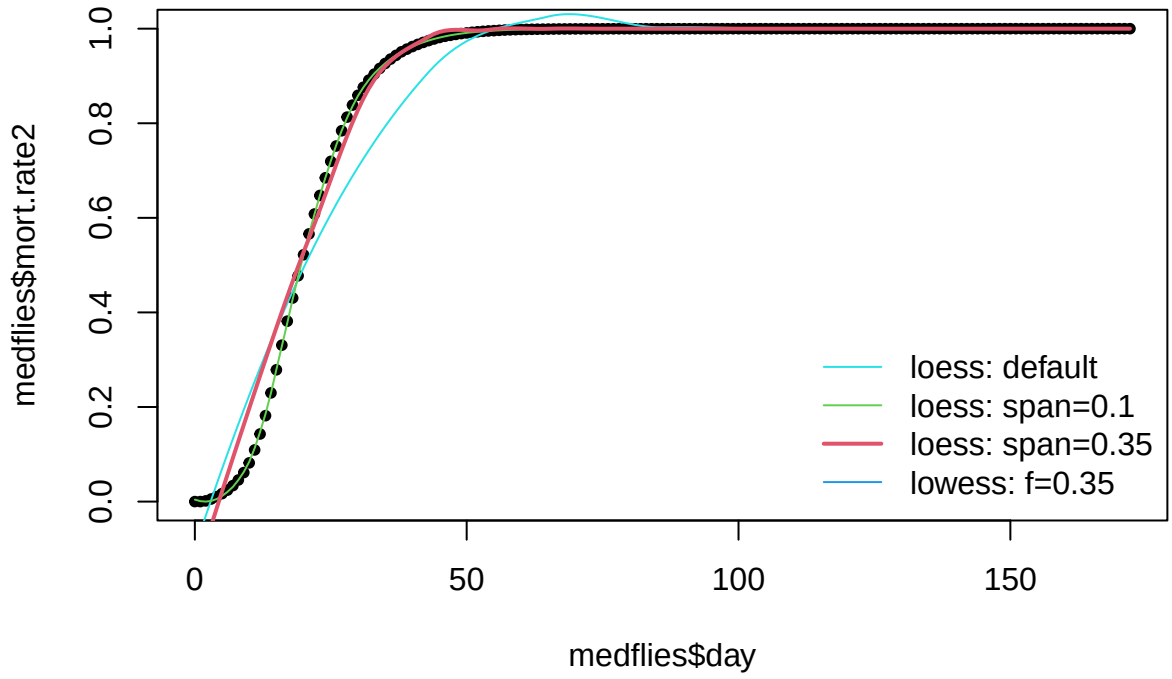
```
plot(medflies$day, medflies$mort.rate2, pch=20)
## If the output of `ksmooth` is not continuous, the default of `x.points`
##  may not be sufficient.
k1 <- ksmooth(medflies$day, medflies$mort.rate2, kernel = "normal", bandwidth=5, x.points=medflies$day)
lines( k1, col=1, lwd = 1)

k2 <- ksmooth(medflies$day, medflies$mort.rate2, kernel = "normal", bandwidth=1, x.points=medflies$day)
lines( k1, col=2, lwd = 1)

k3 <- ksmooth(medflies$day, medflies$mort.rate2, kernel = "box", bandwidth=5, x.points=medflies$day)
lines( k1, col=3, lwd = 1)


k4 <- ksmooth(medflies$day, medflies$mort.rate2, kernel = "box", bandwidth=1, x.points=medflies$day)
lines( k1, col=4, lwd = 1)
```



1.E)

Polynomial smoothing

```
plot(medflies$day, medflies$mort.rate2, pch=20)
lobj <- loess(medflies$mort.rate2 ~ medflies$day)            # default smoothing value
lines(lobj$x, lobj$fitted, col=5)

lobj2 <- loess(medflies$mort.rate2 ~ medflies$day, span=.1)   # not enough smoothing
lines(lobj2$x, lobj2$fitted, col=3)

lobj3 <- loess(medflies$mort.rate2 ~ medflies$day, span=0.35) # probably close to optimal
lines(lobj3$x, lobj3$fitted, col=2, lwd=2)

legend( "bottomright", lty=1, col=c(5,3,2,4), lwd=c(1,1,2), bty='n', legend=
    c('loess: default','loess: span=0.1','loess: span=0.35','lowess: f=0.35'))
```



Splines smoothing

```
dev.off()
```

```
## null device
##           1
```

```
layout(matrix(1:2, 1,2), c(3,1))
(s1 <- smooth.spline(medflies$mort.rate2 ~ medflies$day))      # generalized CV, default
```

```
## Call:
## smooth.spline(x = medflies$mort.rate2 ~ medflies$day)
##
## Smoothing Parameter  spar= 0.1497934  lambda= 2.455804e-08 (14 iterations)
## Equivalent Degrees of Freedom (Df): 77.77004
## Penalized Criterion (RSS): 3.676505e-06
## GCV: 7.013488e-08
```

```
s2 <- smooth.spline(medflies$mort.rate2 ~ medflies$day, cv=T)  # ordinary CV
s3 <- smooth.spline(medflies$mort.rate2 ~ medflies$day, spar=1)
print( c(s1$lambda, s2$lambda, s3$lambda)*1e6)
```

```
## [1] 2.455804e-02 6.129293e-03 3.407670e+04
```

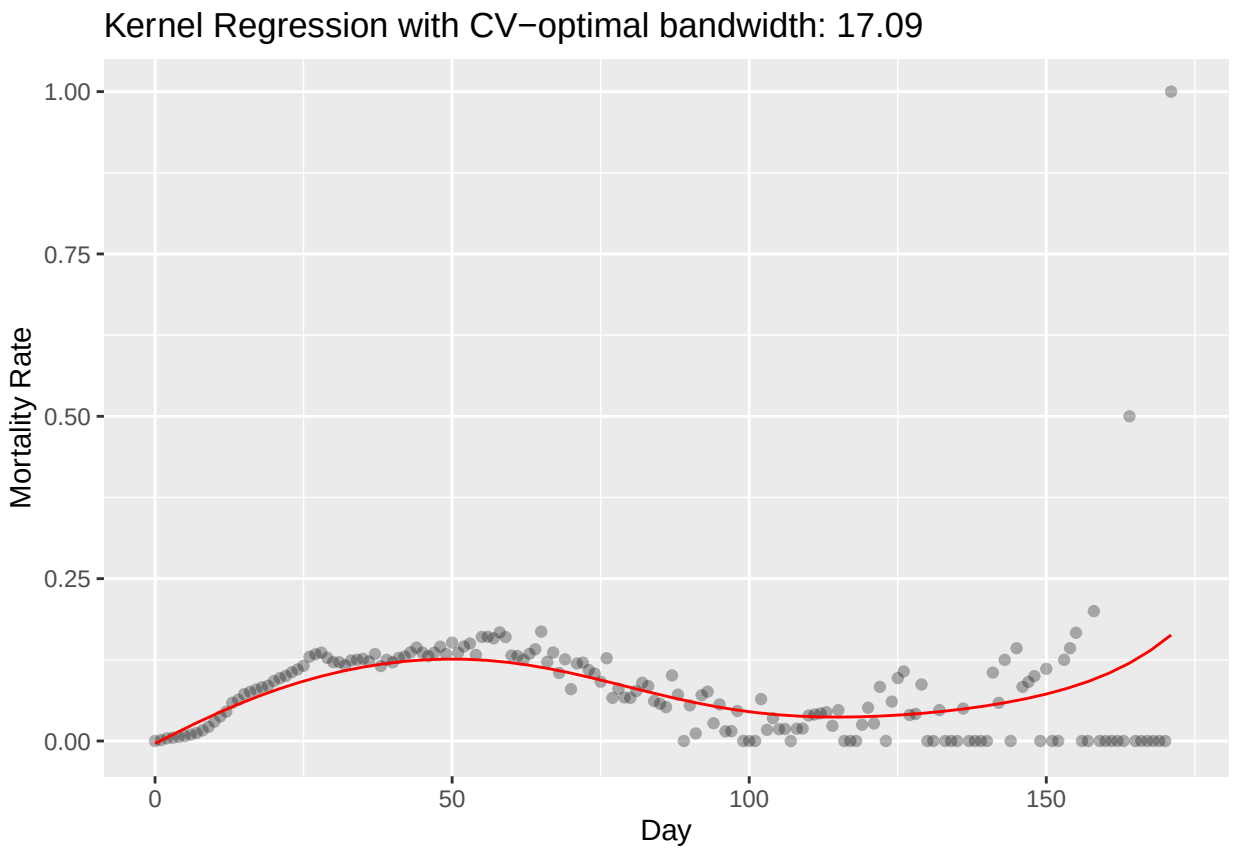
```
plot(medflies$mort.rate2 ~ medflies$day, pch=20)
lines(s1, col=2)
lines(s2, col=3)
lines(s3, col=4, lty=4)
legend("bottomright", legend=c('default (GCV)','CV','spar=1'),
       bty='n', col=c(2,3,4), lty=c(1,1,4))
```

1.F)

```
# Find optimal bandwidth using cross-validation
h.cv <- hcv(medflies$day, medflies$mort.rate)

# Fit model with optimal bandwidth
sm.regression(medflies$day,
              medflies$mort.rate,
              h = h.cv,
              display = "none") -> optimal_fit
```

```
## missing data are removed
```

```
# Plot result
ggplot() +
  geom_point(data = medflies, aes(x = day, y = mort.rate), alpha = 0.3) +
  geom_line(data = data.frame(x = optimal_fit$eval.points,
                              y = optimal_fit$estimate),
            aes(x = x, y = y), color = "red") +
  labs(title = paste("Kernel Regression with CV-optimal bandwidth:", round(h.cv, 2)),
       x = "Day",
       y = "Mortality Rate")
```

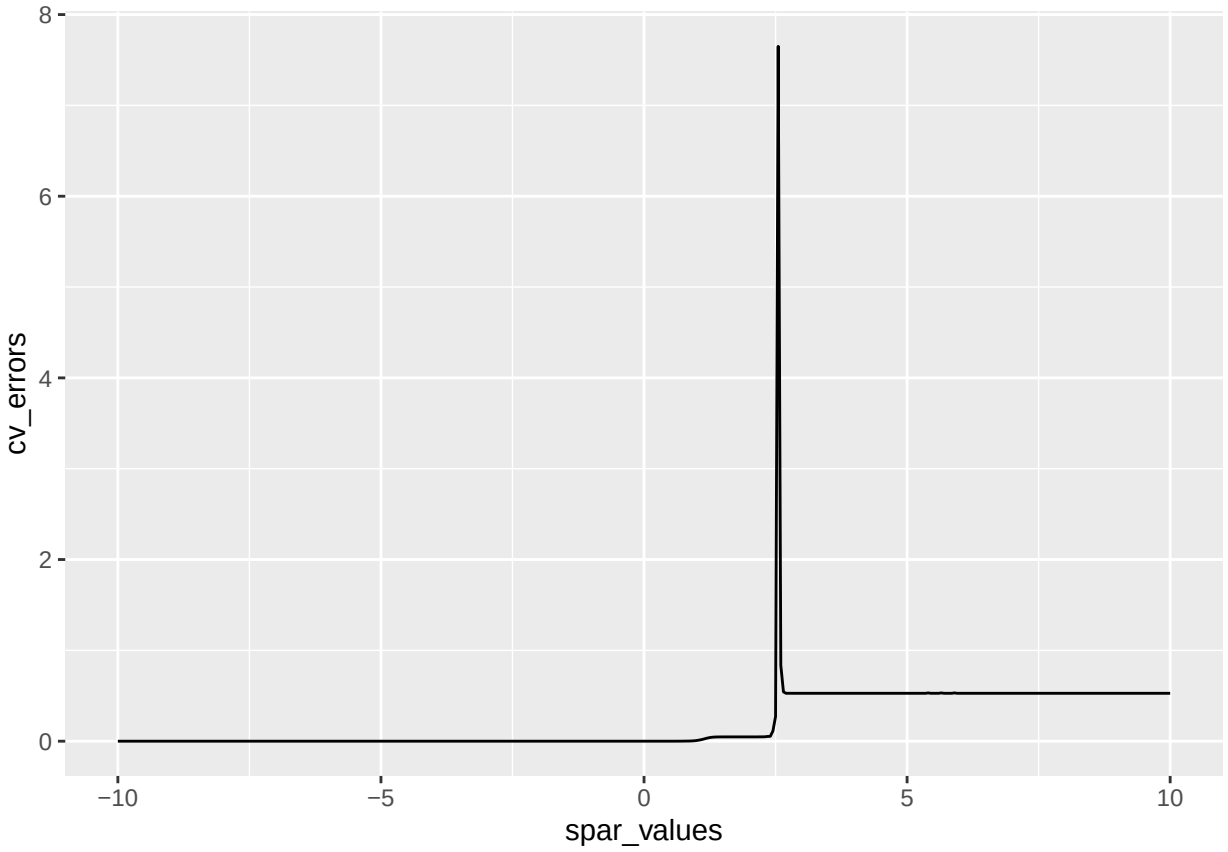


« comments »

The hcv function seems to break down on my machine

1.G)

```r
# Define grid for smoothing parameters
spar_values <- seq(-10, 10, by = 0.05)  # Adjust spar range as needed

# Initialize storage for cross-validation errors
cv_errors <- numeric(length(spar_values))

# Perform cross-validation manually
for (i in seq_along(spar_values)) {
    # Leave-one-out cross-validation for each spar value
    cv_error <- 0

    for (j in 1:nrow(medflies)) {
        # Exclude point j
        train_data <- medflies[-j, ]
        test_data <- medflies[j, ]

        # Fit smoothing spline with specified spar on training data
        spline_fit <- smooth.spline(train_data$day, train_data$mort.rate2, spar = spar_values[i])

        # Predict for left-out point and calculate squared error
        predicted <- predict(spline_fit, test_data$day)$y
        error <- (predicted - test_data$mort.rate2)^2
        cv_error <- cv_error + error
    }

    # Average error for this spar value
    cv_errors[i] <- cv_error / nrow(medflies)
}

# Find optimal spar value
optimal_spar <- spar_values[which.min(cv_errors)]

df <- data.frame(spar_values = spar_values, cv_errors = cv_errors)
df %>% ggplot(aes(x = spar_values, y = cv_errors)) + geom_line()
```




```
cat("Optimal spar value is:", optimal_spar)
```

```
## Optimal spar value is: -0.05
```

1.H)

« comments »

A non-parametric model cannot explain prediction questions out of the range of the data. So if we have data for the first part of an experiment and we want to predcit how the response variable will behave in the future and we know that the normality assumptions are met, we can use a linear regression model.

When the data may have outliers or skewness or we want to avoid assuming a specific underlying distribution for the data we need to focus on medians or ranks instead of means and use non-parametric models. So for example if we are interested in whether there is a significant difference in the affect of two treatments but we are unsure whether the data distribution is normal or not, we can use Wilcoxon rank-sum test.