COS-D407. *Scientific Modeling and Model Validation*

Hands-on excercises

Week 6

University of Helsinki, Finland

26.10.2020–09.12.2020

Lecturer: Christina Bohk-Ewald

Source: https://github.com/christina-bohk-ewald/2020-COS-D407-scientific-modeling-and-model-validation

Table of content:

1. Some preparations in R

1.1 Open a new script for week 6 in R (e.g., *week-6.R*) and save it to a folder of your choice (e.g., *course-COS-D407*).

1.2 Create a filepath to this folder from where you would like to load data and to where you would like to save your outcome. For example,

```
the.course-COS-D407.path <- c("C:/course-COS-D407")
```

1.3 You can then set the working directory to this path

```
setwd(the.course-COS-D407.path)
```

2. Complete sensitivity analysis of COVID-19 demographic scaling model

In week 6 we complete the sensitivity analysis of the demographic scaling model and its COVID-19 infection estimates with respect to changes in IFR_x and D_x. To do this, please run your scripts of week 4 and 5 *with new data as of today* and complete this analysis for estimating COVID-19 infections in Finland based on all 10 combinations of 5 sources for IFR_x and 2 sources for D_x.

2.1 Estimate COVID-19 infections in Finland over time

You can now estimate the total numbers of COVID-19 infections in Finland over time based on all possible combinations of the various IFR_x and D_x estimates, following the steps of the demographic scaling model.

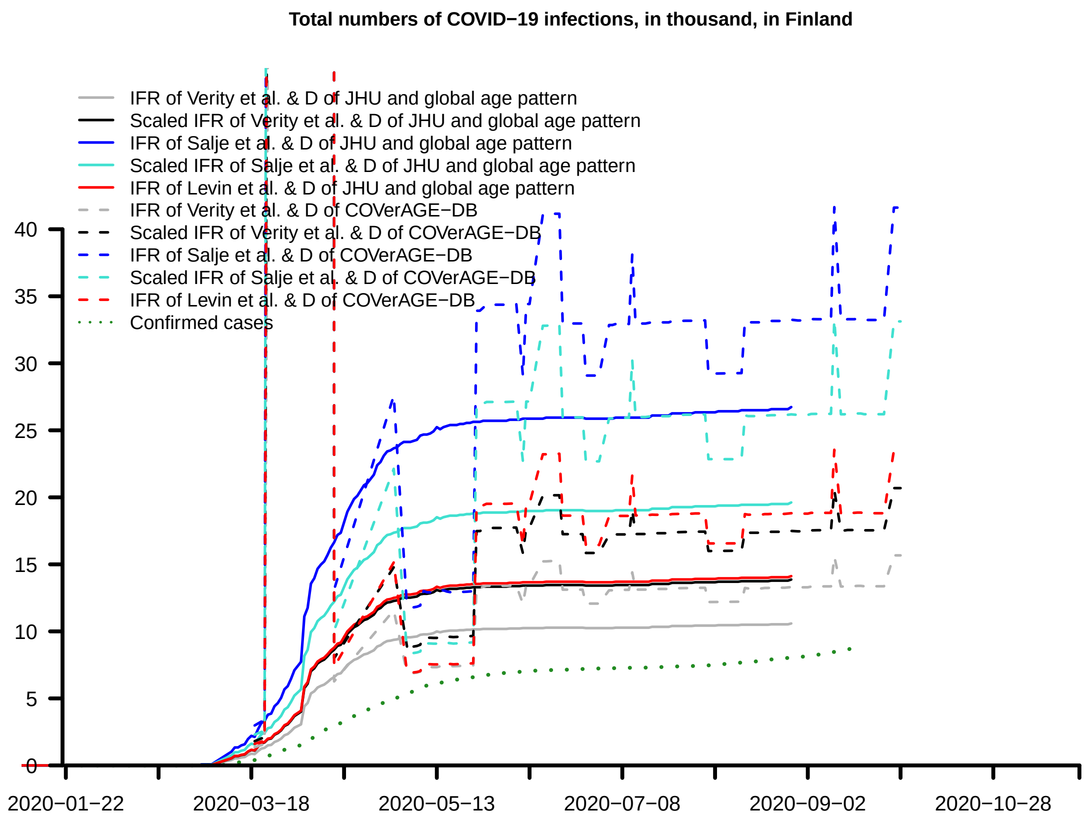
```
#
## Based on data of COVerAGE-DB:
#

inf_mode_scaled_verity_COVerAGE <- colSums( dat_db_10y /
mapped_mode_ifr_thanatAge_verity_10y )

inf_mode_scaled_salje_COVerAGE <- colSums( dat_db_10y /
mapped_mode_ifr_thanatAge_salje_10y )

inf_mode_levin_COVerAGE <- colSums( dat_db_10y /
ifr_by_age_levin )

inf_mode_verity_COVerAGE <- colSums( dat_db_10y /
ifr_by_age_china_verity[,2] )

inf_mode_salje_COVerAGE <- colSums( dat_db_10y /
ifr_by_age_france_salje[,2] )

#
## Based on data of JHU CSSE:
#

inf_mode_scaled_verity_JHU <- colSums( deaths_by_age /
mapped_mode_ifr_thanatAge_verity_10y )

inf_mode_scaled_salje_JHU <- colSums( deaths_by_global_age /
```

```r
mapped_mode_ifr_thanatAge_salje_10y )

inf_mode_levin_JHU <- colSums( deaths_by_global_age /
ifr_by_age_levin )

inf_mode_verity_JHU <- colSums( deaths_by_global_age /
ifr_by_age_china_verity[,2] )

inf_mode_salje_JHU <- colSums( deaths_by_global_age /
ifr_by_age_france_salje[,2] )
```

For example, the data objects *inf_mode_scaled_verity_COVerAGE* and *inf_mode_scaled_verity_JHU* contain the COVID-19 infection estimates for Finland based on the scaled central estimates of the IFR_x for China and the D_x of the COVerAGE-DB and the JHU CSSE (& the global age pattern), respectively.

You can now visualize the total numbers of COVID-19 infections in Finland over time, based on all possible combinations of IFR_x and D_x estimates. When comparing these Finnish COVID-19 estimates with the numbers of confirmed cases, it is a good idea to also account for an average time to death of approximately 18 days.

```r
par(fig = c(0,1,0,1), las=1, mai=c(0.4,0.8,0.8,0.4))

plot(x=-100,y=-100,xlim=c(0,length(dates_all)),ylim=c(0,50),xlab="Date",ylab="",
cex.main=0.9,main="Total numbers of COVID-19 infections, in thousand, in Finland",
axes=FALSE)

## dates_all[which(dates_all %in% dates)]
xxx <- which(dates_all %in% dates)
## dates_all[xxx]

lines(x=xxx-18,y=inf_mode_verity_JHU/1000,lwd=2,col=gray(0.7),lty=1)
lines(x=xxx-18,y=inf_mode_scaled_verity_JHU/1000,lwd=2,col="black",lty=1)
lines(x=xxx-18,y=inf_mode_salje_JHU/1000,lwd=2,col="blue",lty=1)
lines(x=xxx-18,y=inf_mode_scaled_salje_JHU/1000,lwd=2,col="turquoise",lty=1)
lines(x=xxx-18,y=inf_mode_levin_JHU/1000,lwd=2,col="red",lty=1)

lines(x=xxx,y=confirmed[which(confirmed[,"Country.Region"]=="Finland"),
5:ncol(confirmed)]/1000,col="forestgreen",lty=3,lwd=3)

## dates_all[which(dates_all %in% unique(dat$Date))]
xx <- which(dates_all %in% unique(dat$Date))
## dates_all[xx]

lines(x=xx-18,y=inf_mode_verity_COVerAGE/1000,lwd=2,col=gray(0.7),lty=2)
lines(x=xx-18,y=inf_mode_scaled_verity_COVerAGE/1000,lwd=2,col="black",lty=2)
lines(x=xx-18,y=inf_mode_salje_COVerAGE/1000,lwd=2,col="blue",lty=2)
lines(x=xx-18,y=inf_mode_scaled_salje_COVerAGE/1000,lwd=2,col="turquoise",lty=2)
lines(x=xx-18,y=inf_mode_levin_COVerAGE/1000,lwd=2,col="red",lty=2)

axis(side=1,at=c(seq(1,length(dates_all),28),length(dates_all)),
labels=dates_all[c(seq(1,length(dates_all),28),length(dates_all))],lwd=3,pos=0)
axis(side=2,at=seq(0,40,5),labels=TRUE,lwd=3,pos=0)

legend(0,51.5,c("IFR of Verity et al. & D of JHU and global age pattern",
"Scaled IFR of Verity et al. & D of JHU and global age pattern",
"IFR of Salje et al. & D of JHU and global age pattern",
"Scaled IFR of Salje et al. & D of JHU and global age pattern",
"IFR of Levin et al. & D of JHU and global age pattern",
"IFR of Verity et al. & D of COVerAGE-DB",
"Scaled IFR of Verity et al. & D of COVerAGE-DB",
"IFR of Salje et al. & D of COVerAGE-DB",
"Scaled IFR of Salje et al. & D of COVerAGE-DB",
"IFR of Levin et al. & D of COVerAGE-DB",
"Confirmed cases"),
col=c(gray(0.7),"black","blue","turquoise","red",
gray(0.7),"black","blue","turquoise","red","forestgreen"),
bty="n",lwd=2,lty=c(1,1,1,1,1,2,2,2,2,2,3),cex=0.9)
```



Please describe and compare the level and the temporal development of these estimated numbers of COVID-19 infections in Finland. Please also compare them to (1) the numbers of confirmed cases in Finland and, perhaps, to (2) the corresponding figures in other countries.

Please think about how you could extend this R-code in order to estimate the total numbers of COVID-19 infections for other countries and how to provide uncertainty estimates for them.

3. Find suitable model for predicting IFR by age based on *all* raw data

Levin and colleagues (2020) have introduced an exponential model for predicting IFR, in %, by age. As they provide the raw data, you can test how well this exponential model and other models fit these raw data.

3.1 Load raw IFR__x data of Levin et al. (2020)

Please go to the preprint of Levin et al. (2020) on medRxiv and download the Supplementary Data Spreadsheets. You can save this file in your course folder and load the respective data in R:

```
require(openxlsx)

## Representative sample:

levin_ifr_rs <- read.xlsx("LevinEtAl2020-spreadsheet.xlsx",sheet = 2,startRow = 1)
levin_ifr_rs[1:2,]
```

```
##     Study AgeGroup Median_Age Population InfectionRate infrate_ci95_low
## 1 Atlanta     0-17        8.5     402362           0.0              0.0
## 2 Atlanta    18-49       33.5     867468           3.3              1.6
##   infrate_ci95_high Deaths        IFR ifr_ci95_low ifr_ci95_high
## 1               1.0      1 0.00000000   0.00000000            NA
## 2               6.4     20 0.06986547   0.03602438     0.1440975
```

```
## Convenience sample:

levin_ifr_cs <- read.xlsx("LevinEtAl2020-spreadsheet.xlsx",sheet = 3,startRow = 1)
levin_ifr_cs[1:2,]
```

```
##     Study AgeGroup Median_Age Population InfectionRate infrate_ci95_low
## 1 Belgium     0-24         12    3228894      6.686021         4.666667
## 2 Belgium    25-44         35    2956684      6.562703         4.666667
##   infrate_ci95_high Deaths    IFR ifr_ci95_low ifr_ci95_high
## 1          9.555556      1 0.0005       0.0004        0.0007
## 2          9.222222     30 0.0170       0.0120        0.0240
```

```
## Comprehensive tracing:

levin_ifr_ct <- read.xlsx("LevinEtAl2020-spreadsheet.xlsx",sheet = 4,startRow = 1)
levin_ifr_ct[1:2,]
```

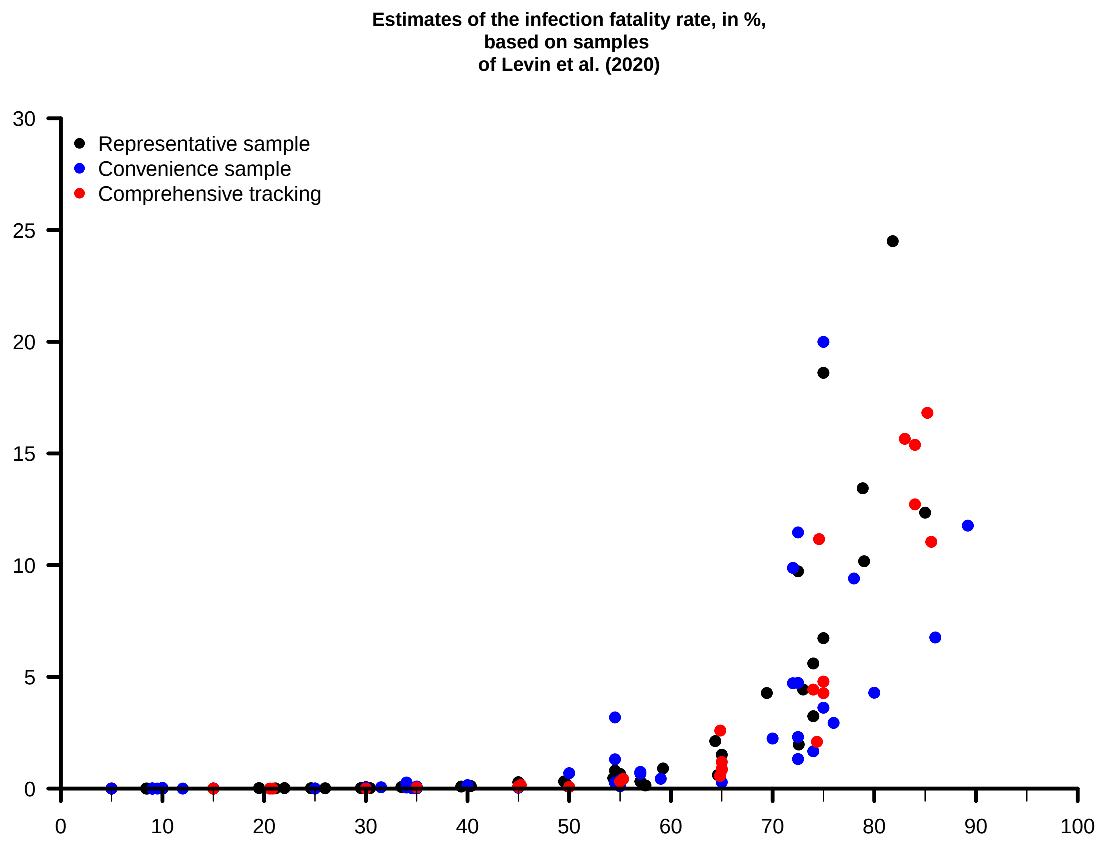
```
##       Study AgeGroup Median_Age Population Infections Deaths InfectionRate
## 1 Australia     0-39   20.55027   13533252       2800      0    0.06206934
## 2 Australia    40-59   50.00000    6413795       2080      3    0.06486020
##   infrate_ci95_low infrate_ci95_high        IFR ifr_ci95_low ifr_ci95_high
## 1       0.03103467        0.09310401 0.00000000   0.00000000            NA
## 2       0.03891612        0.09729029 0.07211538   0.04807692     0.1201923
```

Brief data description. The data objects *levin_ifr_rs*, *levin_ifr_cs*, and *levin_ifr_ct* contain IFR estimates (in%) in column 7 by median age in column 2.

3.2 Plot IFR by median age

To get a feeling for these raw data, you can just plot them.

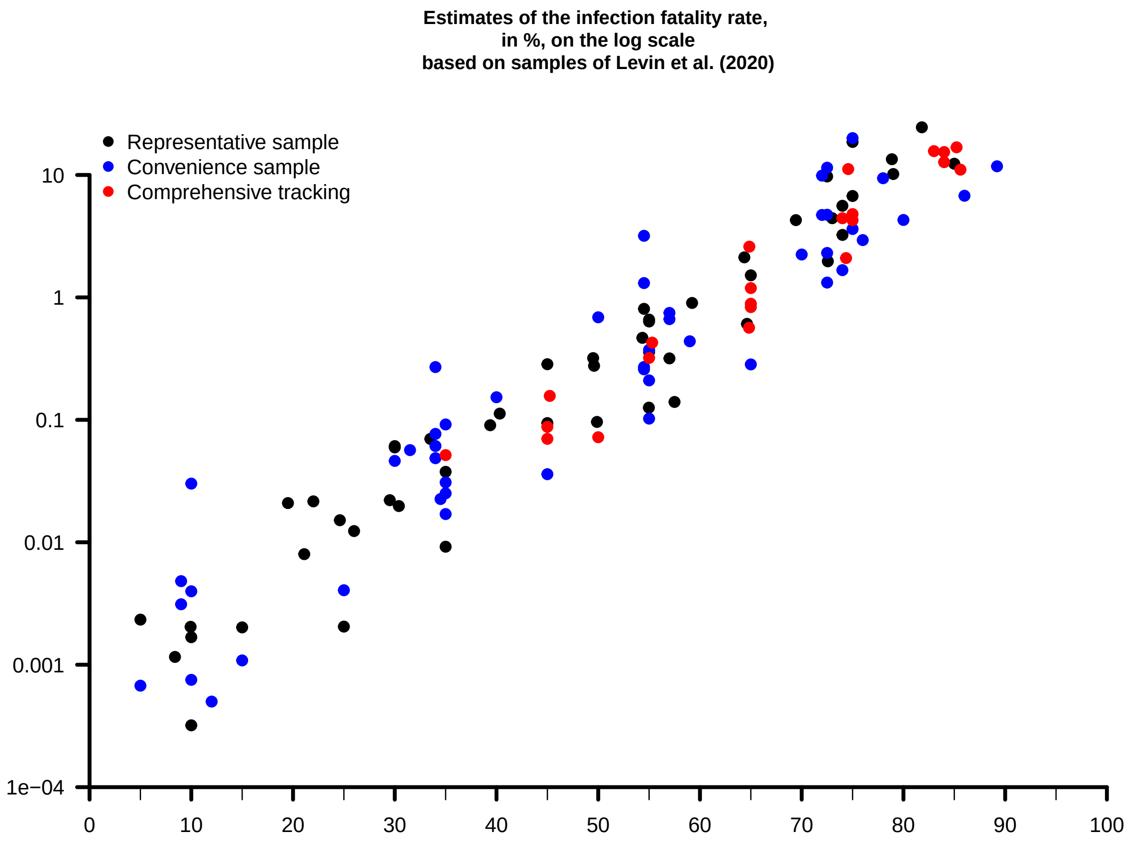
```
par(fig = c(0,1,0,1), las=1, mai=c(0.4,0.8,0.8,0.4))

plot(x=-100,y=-100,xlim=c(0,100),ylim=c(0,30),xlab="Medin age",ylab="",
cex.main=0.9,main="Estimates of the infection fatality rate, in %,\nbased on samples
of Levin et al. (2020)",axes=FALSE)

## names(levin_ifr_rs)
points(x=levin_ifr_rs[,"Median_Age"],y=levin_ifr_rs[,"IFR"],pch=19,col="black",lwd=2)
points(x=levin_ifr_cs[,"Median_Age"],y=levin_ifr_cs[,"IFR"],pch=19,col="blue",lwd=2)
points(x=levin_ifr_ct[,"Median_Age"],y=levin_ifr_ct[,"IFR"],pch=19,col="red",lwd=2)

axis(side=1,at=seq(0,100,5),labels=FALSE,lwd=1,pos=0)
axis(side=1,at=seq(0,100,10),labels=TRUE,lwd=3,pos=0)
axis(side=2,at=seq(0,30,5),labels=TRUE,lwd=3,pos=0)

legend(0,30,c("Representative sample","Convenience sample","Comprehensive tracking"),
pch=rep(19,3),col=c("black","blue","red"),bty="n")
```



. . . and on the logarithmic scale:

```
par(fig = c(0,1,0,1), las=1, mai=c(0.4,0.8,0.8,0.4))

plot(x=-100,y=-100,xlim=c(0,100),ylim=c(log(0.0001,base=exp(1)),log(30,base=exp(1))),
xlab="Medin age",ylab="",cex.main=0.9,main="Estimates of the infection fatality rate,
in %, on the log scale\nbased on samples of Levin et al. (2020)",axes=FALSE)

## names(levin_ifr_rs)
points(x=levin_ifr_rs[,"Median_Age"],y=log(levin_ifr_rs[,"IFR"],base=exp(1)),pch=19,
col="black",lwd=2)

points(x=levin_ifr_cs[,"Median_Age"],y=log(levin_ifr_cs[,"IFR"],base=exp(1)),pch=19,
col="blue",lwd=2)
```

```
points(x=levin_ifr_ct[,"Median_Age"],y=log(levin_ifr_ct[,"IFR"],base=exp(1)),pch=19,
col="red",lwd=2)

axis(side=1,at=seq(0,100,5),labels=FALSE,lwd=1,pos=log(0.0001,base=exp(1)))
axis(side=1,at=seq(0,100,10),labels=TRUE,lwd=3,pos=log(0.0001,base=exp(1)))
axis(side=2,at=log(c(0.0001,0.001,0.01,0.1,1,10),base=exp(1)),
labels=c(0.0001,0.001,0.01,0.1,1,10),lwd=3,pos=0)

legend(0,log(30,base=exp(1)),c("Representative sample","Convenience sample",
"Comprehensive tracking"),pch=rep(19,3),col=c("black","blue","red"),bty="n")
```



3.3 Fit linear and nonlinear models to *all* raw data in order to predict IFR by age

In this section you can fit various models to all raw IFR_x data provided by Levin et al. (2020).

3.3.1 Build data object that contains all raw data In a first step, you can build the data objects *levin_ifr* and *levin_age* that contain all raw data (i.e., 134 data points).

```
levin_ifr <- c(levin_ifr_rs[,"IFR"],levin_ifr_cs[,"IFR"],levin_ifr_ct[,"IFR"])
levin_age <- c(levin_ifr_rs[,"Median_Age"],
            levin_ifr_cs[,"Median_Age"],
            levin_ifr_ct[,"Median_Age"])
```

3.3.2 Fit models 1 through 6 to *all* raw data and analyze their summary statistics In a second step you can fit six (or even more?) models to these raw data that differ in their complexity.

```r
lm_fit <- lm(levin_ifr~levin_age)
## lm_fit
summary(lm_fit)
```

```
##
## Call:
## lm(formula = levin_ifr ~ levin_age)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -4.429 -2.706 -0.883  2.114 17.701
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -3.35483    0.69036  -4.860 3.28e-06 ***
## levin_age    0.12412    0.01295   9.585  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.679 on 132 degrees of freedom
## Multiple R-squared:  0.4104, Adjusted R-squared:  0.4059
## F-statistic: 91.87 on 1 and 132 DF,  p-value: < 2.2e-16
```

```r
lm_fit_2 <- lm(levin_ifr~I(levin_age^2))
## lm_fit_2
summary(lm_fit_2)
```

```
##
## Call:
## lm(formula = levin_ifr ~ I(levin_age^2))
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.9030 -2.3955 -0.0029  1.6351 16.0712
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)    -1.8430110  0.4497300  -4.098 7.23e-05 ***
## I(levin_age^2)  0.0015348  0.0001235  12.424  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.253 on 132 degrees of freedom
## Multiple R-squared:  0.539, Adjusted R-squared:  0.5356
## F-statistic: 154.4 on 1 and 132 DF,  p-value: < 2.2e-16
```

```r
lm_fit_3 <- lm(levin_ifr~I(levin_age^3))
## lm_fit_3
summary(lm_fit_3)
```

```
##
## Call:
## lm(formula = levin_ifr ~ I(levin_age^3))
##
## Residuals:
```

```
##     Min      1Q  Median      3Q     Max
## -5.1039 -1.7190  0.3719  1.1707 14.9290
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)     -1.194e+00  3.609e-01  -3.308  0.00121 **
## I(levin_age^3)   1.966e-05  1.347e-06  14.598  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.963 on 132 degrees of freedom
## Multiple R-squared:  0.6175, Adjusted R-squared:  0.6146
## F-statistic: 213.1 on 1 and 132 DF,  p-value: < 2.2e-16
```

```r
lm_fit_4 <- lm(levin_ifr~I(levin_age^4))
## lm_fit_4
summary(lm_fit_4)
```

```
##
## Call:
## lm(formula = levin_ifr ~ I(levin_age^4))
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.9851 -1.1885  0.3942  0.7453 14.1965
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)     -7.514e-01  3.164e-01  -2.375    0.019 *
## I(levin_age^4)   2.468e-07  1.543e-08  15.998   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.795 on 132 degrees of freedom
## Multiple R-squared:  0.6597, Adjusted R-squared:  0.6572
## F-statistic: 255.9 on 1 and 132 DF,  p-value: < 2.2e-16
```

```r
lm_fit_exp <- lm(log(levin_ifr,base=exp(1))~levin_age, subset=levin_ifr>0)
## lm_fit_exp
summary(lm_fit_exp)
```

```
##
## Call:
## lm(formula = log(levin_ifr, base = exp(1)) ~ levin_age, subset = levin_ifr >
##     0)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.88568 -0.53207  0.03272  0.52574  2.65947
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -7.345377   0.182781  -40.19   <2e-16 ***
## levin_age    0.118387   0.003271   36.20   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.8258 on 119 degrees of freedom
## Multiple R-squared:  0.9167, Adjusted R-squared:  0.916
## F-statistic:  1310 on 1 and 119 DF,  p-value: < 2.2e-16
```

```r
lm_fit_exp_2 <- lm(log(levin_ifr,base=exp(1))~I(levin_age^2), subset=levin_ifr>0)
## lm_fit_exp_2
summary(lm_fit_exp_2)
```

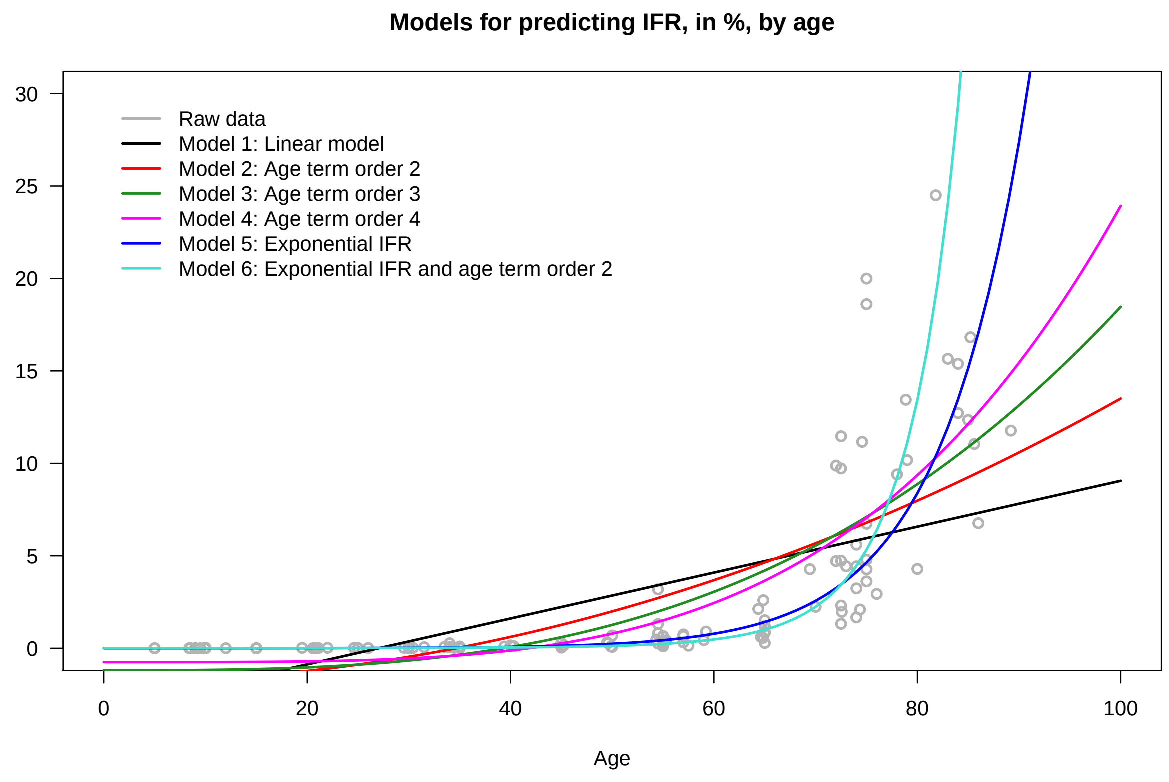
```
##
## Call:
## lm(formula = log(levin_ifr, base = exp(1)) ~ I(levin_age^2),
##     subset = levin_ifr > 0)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -3.12701 -0.63996  0.09893  0.78511  2.65380
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)    -5.039e+00  1.632e-01  -30.87   <2e-16 ***
## I(levin_age^2)  1.193e-03  4.263e-05   27.99   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.039 on 119 degrees of freedom
## Multiple R-squared:  0.8681, Adjusted R-squared:  0.867
## F-statistic: 783.5 on 1 and 119 DF,  p-value: < 2.2e-16
```

You can compare how well these models fit *all* raw data using standard diagnostic tools that R provides, e.g., via the function *summary()*. What do you think: what is the most suitable model for predicting IFR by age based on these summary statistics?

Something to think about. You could also extend this analysis and add a model of your choice here.

3.3.3 Plot how well linear and nonlinear models fit raw data of IFR by age In a third step you can visualize how well the six (or more?) models fit the raw data:

```r
par(fig = c(0,1,0,1), las=1, mai=c(1.4,0.8,0.8,0.4))

plot(levin_age,levin_ifr,col=gray(0.7),xlim=c(0,100),ylim=c(0,30),lwd=2, xlab="Age",
ylab="",main="Models for predicting IFR, in %, by age")
lines(x=seq(0,100,1),y=predict(lm_fit,data.frame(levin_age=(seq(0,100,1)))),lwd=2)
lines(x=seq(0,100,1),y=predict(lm_fit_2,data.frame(levin_age=(seq(0,100,1)))),
col="red",lwd=2)
lines(x=seq(0,100,1),y=predict(lm_fit_3,data.frame(levin_age=(seq(0,100,1)))),
col="forestgreen",lwd=2)
lines(x=seq(0,100,1),y=predict(lm_fit_4,data.frame(levin_age=(seq(0,100,1)))),
col="magenta",lwd=2)
lines(x=seq(0,100,1),y=exp(predict(lm_fit_exp,data.frame(levin_age=(seq(0,100,1))))),
col="blue",lwd=2)
lines(x=seq(0,100,1),y=exp(predict(lm_fit_exp_2,data.frame(levin_age=(seq(0,100,1))))),
col="turquoise",lwd=2)
legend(0,30, c("Raw data","Model 1: Linear model","Model 2: Age term order 2",
"Model 3: Age term order 3","Model 4: Age term order 4","Model 5: Exponential IFR",
"Model 6: Exponential IFR and age term order 2"),
col=c(gray(0.7),"black","red","forestgreen","magenta","blue","turquoise"),lwd=2,bty="n")
```



Again, based on these new information, what do you think: what is the most suitable model for predicting IFR by age?

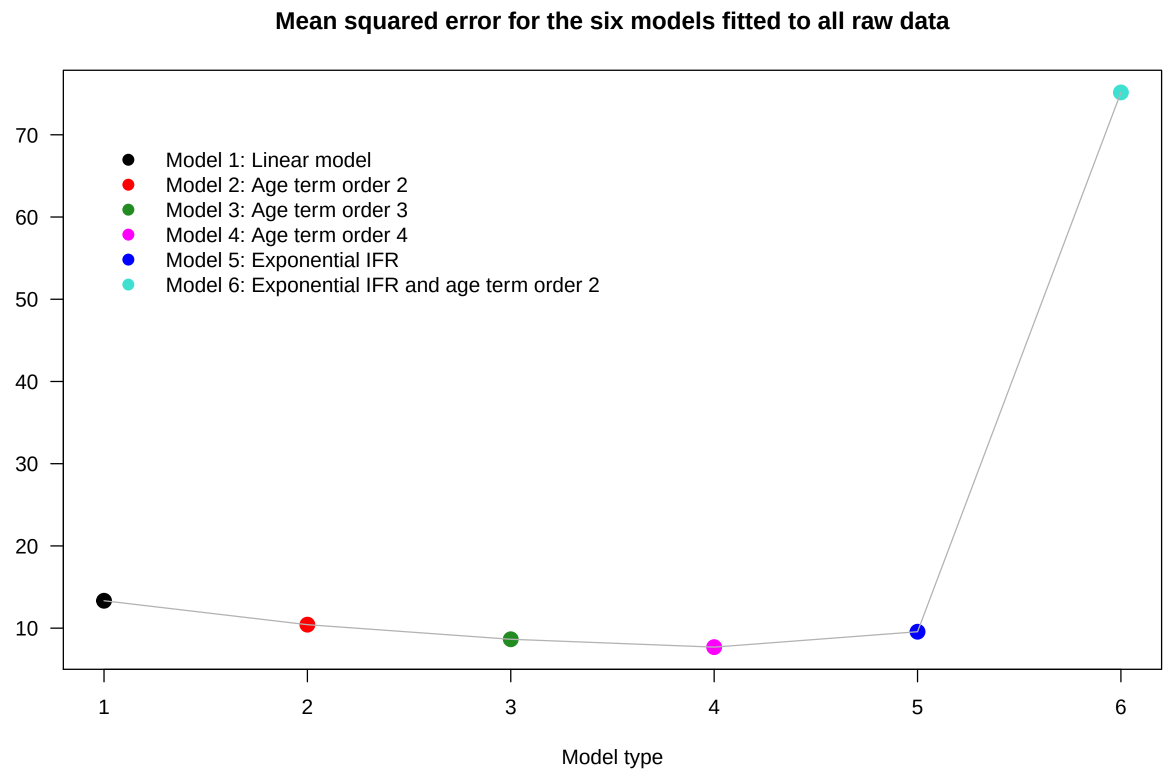
3.3.4 Alternative metric to assess model performance: the mean squared error As an alternative to the statistics provided in the R function *summary()*, you can also compute the mean squared error (MSE) between the fitted and raw data:

```
mse_all_data <- c(mean((levin_ifr - predict(lm_fit,data.frame(levin_age=levin_age)))^2),
            mean((levin_ifr - predict(lm_fit_2,data.frame(levin_age=levin_age)))^2),
            mean((levin_ifr - predict(lm_fit_3,data.frame(levin_age=levin_age)))^2),
            mean((levin_ifr - predict(lm_fit_4,data.frame(levin_age=levin_age)))^2),
            mean((levin_ifr - exp(predict(lm_fit_exp,data.frame(levin_age=levin_age))))^2),
            mean((levin_ifr - exp(predict(lm_fit_exp_2,data.frame(levin_age=levin_age))))^2))
```

And then you can, of course, plot the mean MSE in order to compare how well the six (or even more?) models fit *all* raw data and predict IFR by age:

```
par(fig = c(0,1,0,1), las=1, mai=c(1.4,0.8,0.8,0.4))

plot(1:6,mse_all_data,col=c("black","red","forestgreen","magenta","blue","turquoise"),
pch=19,lwd=5,xlab="Model type",ylab="",
main="Mean squared error for the six models fitted to all raw data")
lines(1:6,mse_all_data,col=gray(0.7))
```

```r
legend(1,70, c("Model 1: Linear model","Model 2: Age term order 2",
"Model 3: Age term order 3","Model 4: Age term order 4",
"Model 5: Exponential IFR","Model 6: Exponential IFR and age term order 2"),
col=c("black","red","forestgreen","magenta","blue","turquoise"),pch=rep(19,6),
lty=NA,lwd=2,bty="n")
```



Which of these six (or more?) models has the smallest average MSE? How does this compare to the summary statistics of the six models in section 3.3.2? Based on these new information, what do you think: what is the most suitable method to predict IFR by age?

Something more to think about. What could possibly be wrong with fitting and evaluating the six models for predicting IFR by age using *all* raw data?

4. Find suitable model for predicting IFR by age based on training data and testing data

Thinking critically about the procedure adopted in section 3, it might not be so much about finding the model that fits best to *all* raw data, but rather about finding the model that predicts best values of the IFR by age the model has not seen yet (machine learning; generalization of underlying pattern). That is why we redo (or refine) the analysis above and split *all* raw data into training data and testing data in order to find the model that predicts best IFR by age. Following this approach, we use the training data to fit the models and we use the testing data to test how well the models predict IFR by age in a *new* environment.

4.1 Validation set approach

According to the validation set approach, you randomly split *all* raw data into two parts of (roughly) equal size that are the training data and the testing data. You then fit the six models to the training data before you compare their IFR predictions with the actual IFR values of the testing data using the MSE.

```r
train_MSE <- matrix(NA,nr=6,nc=10)
test_MSE <- matrix(NA,nr=6,nc=10)

for(run in 1:10){

    ## 4.1.1 Randomly split all data into training and testing data:

    set.seed(run)
    train <- sample(length(levin_ifr),length(levin_ifr)/2)
    test <- c(1:length(levin_ifr))[-train]
    ## sort(train)
    ## sort(test)

    train_levin_ifr <- levin_ifr[train]
    train_levin_age <- levin_age[train]
    test_levin_ifr <- levin_ifr[test]
    test_levin_age <- levin_age[test]

    ## 4.1.2 Fit models to training data:

    model_1 <- lm(train_levin_ifr~train_levin_age)
    model_2 <- lm(train_levin_ifr~I(train_levin_age^2))
    model_3 <- lm(train_levin_ifr~I(train_levin_age^3))
    model_4 <- lm(train_levin_ifr~I(train_levin_age^4))
    model_5 <- lm(log(train_levin_ifr,base=exp(1))~train_levin_age,
subset=train_levin_ifr>0)

    model_6 <- lm(log(train_levin_ifr,base=exp(1))~I(train_levin_age^2),
subset=train_levin_ifr>0)

    ## 4.1.3 Calculate and save train MSE:

    train_MSE[1,run] <- mean((train_levin_ifr -
predict(model_1,data.frame(train_levin_age=train_levin_age)))^2)

    train_MSE[2,run] <- mean((train_levin_ifr -
predict(model_2,data.frame(train_levin_age=train_levin_age)))^2)

    train_MSE[3,run] <- mean((train_levin_ifr -
predict(model_3,data.frame(train_levin_age=train_levin_age)))^2)

    train_MSE[4,run] <- mean((train_levin_ifr -
predict(model_4,data.frame(train_levin_age=train_levin_age)))^2)

    train_MSE[5,run] <- mean((train_levin_ifr -
exp(predict(model_5,data.frame(train_levin_age=train_levin_age))))^2)

    train_MSE[6,run] <- mean((train_levin_ifr -
exp(predict(model_6,data.frame(train_levin_age=train_levin_age))))^2)
```

```r
    ## 4.1.4 Apply fitted models to testing data and calculate and save test MSE:

    test_MSE[1,run] <- mean((test_levin_ifr -
predict(model_1,data.frame(train_levin_age=test_levin_age)))^2)

    test_MSE[2,run] <- mean((test_levin_ifr -
predict(model_2,data.frame(train_levin_age=test_levin_age)))^2)

    test_MSE[3,run] <- mean((test_levin_ifr -
predict(model_3,data.frame(train_levin_age=test_levin_age)))^2)

    test_MSE[4,run] <- mean((test_levin_ifr -
predict(model_4,data.frame(train_levin_age=test_levin_age)))^2)

    test_MSE[5,run] <- mean((test_levin_ifr -
exp(predict(model_5,data.frame(train_levin_age=test_levin_age))))^2)

    test_MSE[6,run] <- mean((test_levin_ifr -
exp(predict(model_6,data.frame(train_levin_age=test_levin_age))))^2)

} ## for run
```

The R-code above applies the validation set approach ten times (run 1 through ten in the for loop). The data objects *train_MSE* and *test_MSE* contain the MSE for each of the six models (rows) in each of the 10 runs (columns) when applied to the training data and the testing data, respectively.

These data allow you to eventually compute the average train MSE and the average test MSE across all ten runs for each of the six models:

```r
rowMeans(train_MSE)
```

```
## [1] 12.955992  9.899420  8.036573  7.022064  8.565164 55.786678
```

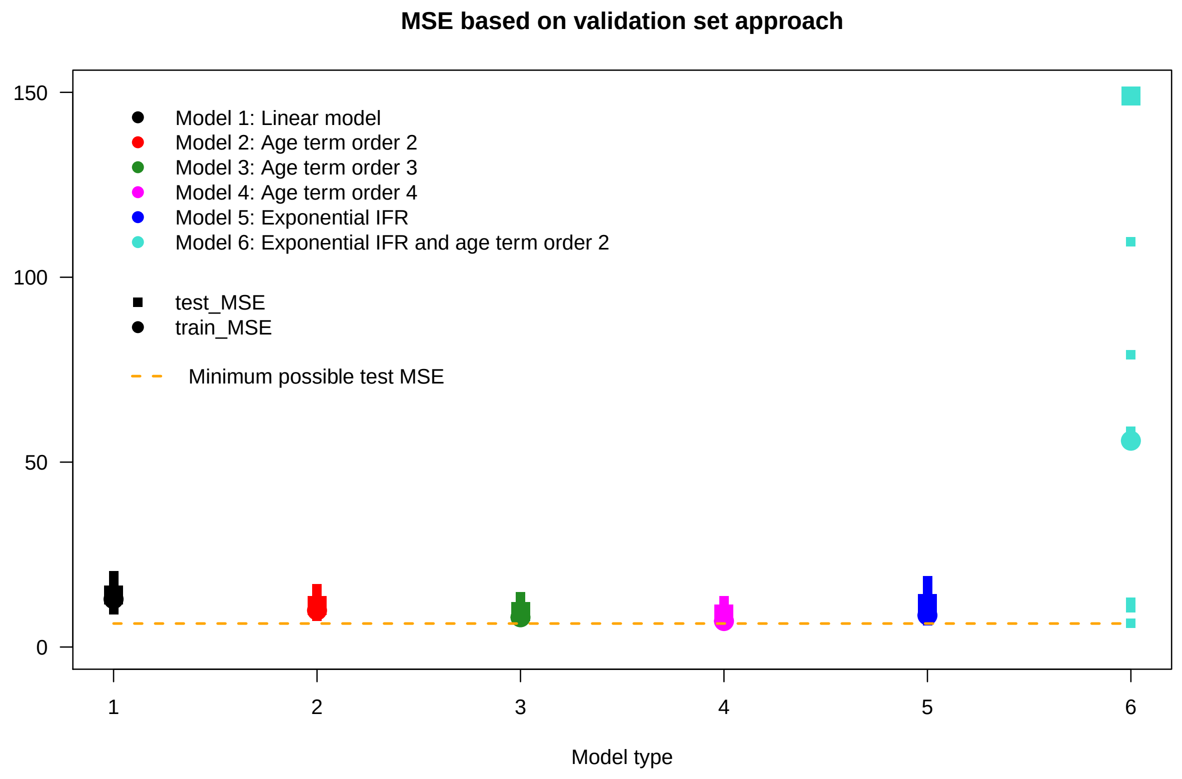
```r
rowMeans(test_MSE)
```

```
## [1]  14.003603  11.279971   9.657318   8.882602  11.759862 149.008199
```

Which model has the smallest / largest testing MSE? Which model has the smallest / largest training MSE? You can also plot your findings:

```r
par(fig = c(0,1,0,1), las=1, mai=c(1.4,0.8,0.8,0.4))

plot(1:6,rowMeans(train_MSE),main="MSE based on validation set approach",
col=c("black","red","forestgreen","magenta","blue","turquoise"),
pch=19,cex=2,xlab="Model type",ylab="",ylim=c(0,150))

points(1:6,rowMeans(test_MSE),
col=c("black","red","forestgreen","magenta","blue","turquoise"),pch=15,cex=2)

for(model in 1:6){
    points(x=rep(model,10),y=test_MSE[model,],
col=c("black","red","forestgreen","magenta","blue","turquoise")[model],pch=15,cex=1)
}

lines(1:6,rep(min(test_MSE),6),col="orange",lty=2,lwd=2)
```

```
legend(1,150, c("Model 1: Linear model","Model 2: Age term order 2",
"Model 3: Age term order 3","Model 4: Age term order 4",
"Model 5: Exponential IFR","Model 6: Exponential IFR and age term order 2"),
col=c("black","red","forestgreen","magenta","blue","turquoise"),
pch=rep(19,6),lty=NA,lwd=2,bty="n")

legend(1,100,col=c("black","black"),c("test_MSE","train_MSE"),
lty=NA,pch=c(15,19),lwd=2,bty="n")

legend(1,80,col=c("orange"),c("Minimum possible test MSE"),lty=c(2),lwd=2,bty="n")
```

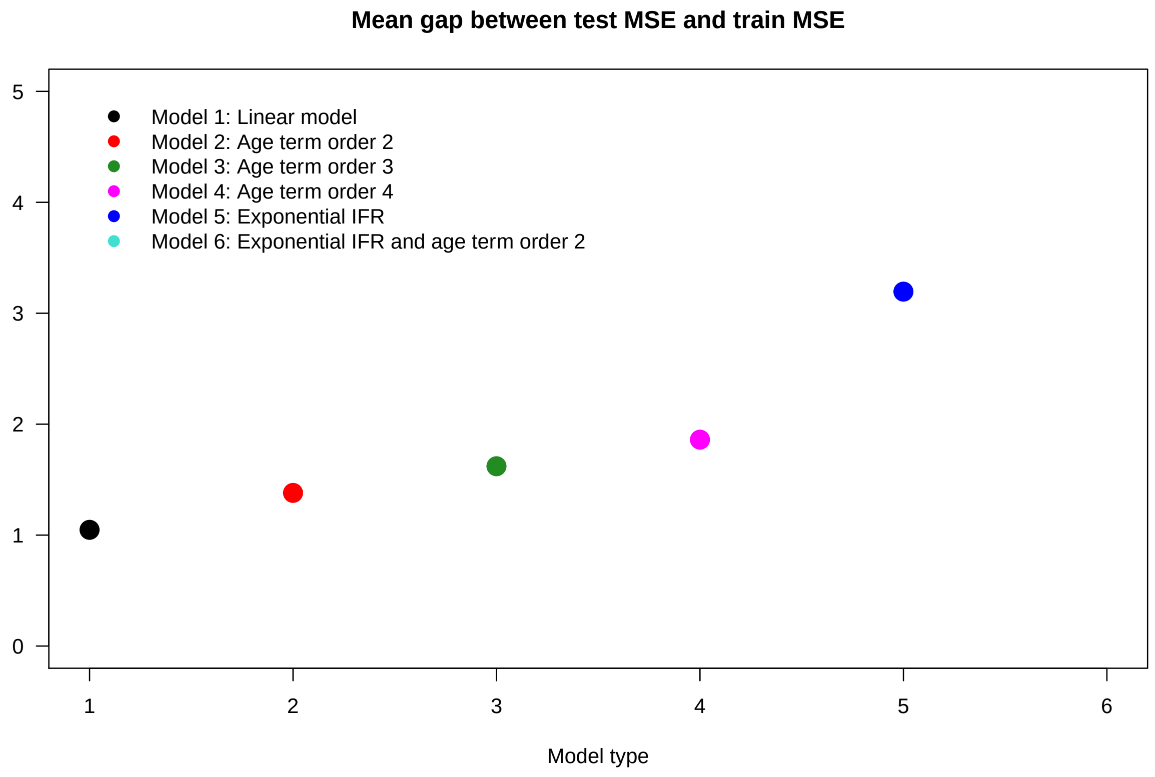


Something to think about. What do these information tell you about the suitability of each of these models for predicting the IFR by age? What model(s) have comparably low bias and low variance?

To answer this question it might be convenient to zoom into the plot above and it might also be interesting to visualize the gap between average test MSE and average train MSE:

```
gap <- test_MSE-train_MSE

par(fig = c(0,1,0,1), las=1, mai=c(1.4,0.8,0.8,0.4))

plot(1:6,rowMeans(gap),main="Mean gap between test MSE and train MSE",
col=c("black","red","forestgreen","magenta","blue","turquoise"),
pch=19,cex=2,xlab="Model type",ylab="",ylim=c(0,5))

lines(1:6,rep(min(gap),6),col="orange",lty=2,lwd=2)
```

```r
legend(1,5, c("Model 1: Linear model","Model 2: Age term order 2",
"Model 3: Age term order 3","Model 4: Age term order 4",
"Model 5: Exponential IFR","Model 6: Exponential IFR and age term order 2"),
col=c("black","red","forestgreen","magenta","blue","turquoise"),pch=rep(19,6),
lty=NA,lwd=2,bty="n")
```



Something more to think about. Why can low bias and low variance be regarded as indicators for a suitable model? What could limitate your findings based on the validation set approach, e.g., given that *all* raw data consist of only 134 data points?

5. Time for you to think both creatively and critically about selecting the most suitable method for predicting IFR by age based on raw data provided by Levin and colleagues (2020).

You can consider here the pros and cons of the different procedures adopted to fit and evaluate the models with respect to their ability to accurately predcit IFR by age. For example, what is the actual purpose of a model that has been designed to predict IFR by age? What does it mean when a model is fitted to training data instead of to *all* raw data? And what does it mean when a model's predictions are evaluated using testing data instead of *all* raw data?

When selecting the most suitable model (among those six models) for predicting the IFR by age, you may also want to consider the theoretical meaning of the models; e.g., to distinguish between M4 and M5?