Kolmogorov-Smirnov Test Regression

Dean Gladish and Nick Reich

July 1st, 2018

A Base Case for the Previous Kolmogorov-Smirnov Test

The KCDE and KDE Models' predictions for bins

Let's say that we are looking at the KDE model, so named because it uses Kernel Density Estimation to measure a distribution for each target. We want to compare the probabilities that it assigns to each bin (each interval for the Influenza-Like Illness (ILI) outcome) with another model such as the KCDE model. The KCDE model uses Kernel Conditional Density Estimation to define a distribution for ILI based on recent observations of ILI and the current week of the flu season.

```r
library(Sleuth3)
library(dplyr)
library(ggformula)
library(pander)
library(knitr)
library(stargazer)
library(car)
library(pander)
library(gridExtra)
library(broom)
library(ggthemes)
library(MASS)
library(leaps)
library(GGally)
library(effects)
library(grid)
library(grImport2)
library(ggplot2)
# In addition to generating four plots of K-S statistics
# we now want to show how the difference in distributions might
# depend on the ILI level (we say that the ILI level is
# given in each .csv file by the Value of the
# single row of type Point (as opposed to Bin)
# that is included for each section  called
# 1 wk ahead, 2 wk ahead, etc.)
USNationalXWeeksAhead <- function(dataset, week, pointPredictionforILI) {
  # Removes all entries where Location is not US National
  dataset <- dataset[!(dataset$Location != "US National"),]

  if (pointPredictionforILI == "no") {
    # Removes all entries where Type is not Bin
    dataset <- dataset[!(dataset$Type != "Bin"),]
  }
  if (pointPredictionforILI == "yes") {
    # Removes all entries where Type is not Point.
    dataset <- dataset[!(dataset$Type != "Point"),]
  }

  # Removes all entries that do not refer to the model's
  # probabilities assigned to the ILI bins for the specified
  # number of weeks ahead.
  if (week == "week one") {
    dataset <- dataset[!(dataset$Target != "1 wk ahead"),]
  }
  if (week == "week two") {
    dataset <- dataset[!(dataset$Target != "2 wk ahead"),]
  }
  if (week == "week three") {
    dataset <- dataset[!(dataset$Target != "3 wk ahead"),]
  }
  if (week == "week four") {
    dataset <- dataset[!(dataset$Target != "4 wk ahead"),]
  }
  return(dataset)
}



# We take each of the files for the kcde model
kcdeFiles <- lapply(Sys.glob(paste("C:/Users/gladi/Documents/GitHu",
                                   "b/cdc-flusight-ensemble/model-f",
                                   "orecasts/real-time-component-mod",
                                   "els/ReichLab_kcde/*.csv", sep = "")),
                    read.csv)


kdeFiles <- lapply(Sys.glob(paste("C:/Users/gladi/Document",
                                  "s/GitHub/cdc-flusight-ens",
                                  "emble/model-forecasts/real",
                                  "-time-component-models/Rei",
                                  "chLab_kde/*.csv", sep = "")),
                   read.csv)


# First we should look at the values:
summary(kcdeFiles[[1]]$Value)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##  0.0000  0.0000  0.0005  0.1275  0.0091 52.0000
```

```
summary(kdeFiles[[1]]$Value)
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max. 
##  0.00001  0.00047  0.00347  0.08380  0.01414 52.00000
```

```
# It seems like the kcde model has quite a larger maximum.

# We also want to create four copies of each file, each of which
# will be refined to focus on a specific week (1, 2, 3, 4).
```

```r
kcdeWeek1 <- kcdeFiles
kcdeWeek2 <- kcdeFiles
kcdeWeek3 <- kcdeFiles
kcdeWeek4 <- kcdeFiles
kcdeWeek1pointPredictionforILI <- kcdeFiles
kcdeWeek2pointPredictionforILI <- kcdeFiles
kcdeWeek3pointPredictionforILI <- kcdeFiles
kcdeWeek4pointPredictionforILI <- kcdeFiles


kdeWeek1 <- kdeFiles
kdeWeek2 <- kdeFiles
kdeWeek3 <- kdeFiles
kdeWeek4 <- kdeFiles
kdeWeek1pointPredictionforILI <- kdeFiles
kdeWeek2pointPredictionforILI <- kdeFiles
kdeWeek3pointPredictionforILI <- kdeFiles
kdeWeek4pointPredictionforILI <- kdeFiles

for (i in 1:length(kcdeFiles)) {
  # We have taken each file for each week of the KCDE model
  # and would like to look at the predictions for each of
  # 1, 2, 3, and 4 weeks ahead.
  kcdeWeek1[[i]] <- USNationalXWeeksAhead(kcdeFiles[[i]], "week one", "no")
  kcdeWeek2[[i]] <- USNationalXWeeksAhead(kcdeFiles[[i]], "week two", "no")
  kcdeWeek3[[i]] <- USNationalXWeeksAhead(kcdeFiles[[i]], "week three", "no")
  kcdeWeek4[[i]] <- USNationalXWeeksAhead(kcdeFiles[[i]], "week four", "no")

  kcdeWeek1pointPredictionforILI[[i]] <- USNationalXWeeksAhead(kcdeFiles[[i]],
                                                               "week one", "yes")
  kcdeWeek2pointPredictionforILI[[i]] <- USNationalXWeeksAhead(kcdeFiles[[i]],
                                                               "week two", "yes")
  kcdeWeek3pointPredictionforILI[[i]] <- USNationalXWeeksAhead(kcdeFiles[[i]],
                                                               "week three", "yes")
  kcdeWeek4pointPredictionforILI[[i]] <- USNationalXWeeksAhead(kcdeFiles[[i]],
                                                               "week four", "yes")
}

for (i in 1:length(kdeFiles)) {
  # We should also do this for the KDE model.
  kdeWeek1[[i]] <- USNationalXWeeksAhead(kdeFiles[[i]], "week one", "no")
  kdeWeek2[[i]] <- USNationalXWeeksAhead(kdeFiles[[i]], "week two", "no")
  kdeWeek3[[i]] <- USNationalXWeeksAhead(kdeFiles[[i]], "week three", "no")
  kdeWeek4[[i]] <- USNationalXWeeksAhead(kdeFiles[[i]], "week four", "no")

  kdeWeek1pointPredictionforILI[[i]] <- USNationalXWeeksAhead(kdeFiles[[i]],
                                                              "week one", "yes")
  kdeWeek2pointPredictionforILI[[i]] <- USNationalXWeeksAhead(kdeFiles[[i]],
                                                              "week two", "yes")
  kdeWeek3pointPredictionforILI[[i]] <- USNationalXWeeksAhead(kdeFiles[[i]],
                                                              "week three", "yes")
  kdeWeek4pointPredictionforILI[[i]] <- USNationalXWeeksAhead(kdeFiles[[i]],
                                                              "week four", "yes")
}

# We will need to install and load a specific package called rlist
# in order to remove specific parts of the kdeWeekX files because
# they contain files beginning at four weeks that are not included
# in the kcdeWeekX files.

# We have to remove weeks 19, 40, 41, and 42 since these are skipped
# in the kcde files.

# After installing this package we load it:

library(rlist)

# The package gives us the list.remove() function.

length(kdeWeek1)
```

```
## [1] 32
```

```
removeStuff <- function(list) {
  # Removes elements 19, 20, 21, and 22.
  return(list.remove(list, c(19, 20, 21, 22)))
  # Since R works this way we can just do it all at once
}

kdeWeek1Reduced <- removeStuff(kdeWeek1)
kdeWeek2Reduced <- removeStuff(kdeWeek2)
kdeWeek3Reduced <- removeStuff(kdeWeek3)
kdeWeek4Reduced <- removeStuff(kdeWeek4)

kdeWeek1pointPredictionforILIReduced <- removeStuff(kdeWeek1pointPredictionforILI)
kdeWeek2pointPredictionforILIReduced <- removeStuff(kdeWeek2pointPredictionforILI)
kdeWeek3pointPredictionforILIReduced <- removeStuff(kdeWeek3pointPredictionforILI)
kdeWeek4pointPredictionforILIReduced <- removeStuff(kdeWeek4pointPredictionforILI)
# Here, all but the rows we want to look at are removed.
```

There is one crucial thing that we need to do as well before generating the plots: Because our files only share their respective data until week 18 of 2018 and then refer back to the last few weeks of 2017, we need to rotate the order of our files within each of our 8 files.

```
# We will just do another for-loop to replace these values and
# achieve what is functionally a rotation.
temp1 <- kcdeWeek1
temp2 <- kcdeWeek2
temp3 <- kcdeWeek3
temp4 <- kcdeWeek4
temp5 <- kdeWeek1Reduced
temp6 <- kdeWeek2Reduced
temp7 <- kdeWeek3Reduced
temp8 <- kdeWeek4Reduced

# We also need to do this for our ILI values.
temp9 <- kcdeWeek1pointPredictionforILI
temp10 <- kcdeWeek2pointPredictionforILI
```

```r
temp11 <- kcdeWeek3pointPredictionforILI
temp12 <- kcdeWeek4pointPredictionforILI
temp13 <- kdeWeek1pointPredictionforILIReduced
temp14 <- kdeWeek2pointPredictionforILIReduced
temp15 <- kdeWeek3pointPredictionforILIReduced
temp16 <- kdeWeek4pointPredictionforILIReduced

for (i in 1:28) {
  # The modular portion (taking the remainder)
  # ensures that we do not go outside the bounds of
  # possible indices.
  kcdeWeek1[[i]] <- temp1[[((i+17)%%28)+1]]
  kcdeWeek2[[i]] <- temp2[[((i+17)%%28)+1]]
  kcdeWeek3[[i]] <- temp3[[((i+17)%%28)+1]]
  kcdeWeek4[[i]] <- temp4[[((i+17)%%28)+1]]

  kdeWeek1Reduced[[i]] <- temp5[[((i+17)%%28)+1]]
  kdeWeek2Reduced[[i]] <- temp6[[((i+17)%%28)+1]]
  kdeWeek3Reduced[[i]] <- temp7[[((i+17)%%28)+1]]
  kdeWeek4Reduced[[i]] <- temp8[[((i+17)%%28)+1]]

  kcdeWeek1pointPredictionforILI[[i]] <- temp9[[((i+17)%%28)+1]]
  kcdeWeek2pointPredictionforILI[[i]] <- temp10[[((i+17)%%28)+1]]
  kcdeWeek3pointPredictionforILI[[i]] <- temp11[[((i+17)%%28)+1]]
  kcdeWeek4pointPredictionforILI[[i]] <- temp12[[((i+17)%%28)+1]]

  kdeWeek1pointPredictionforILIReduced[[i]] <- temp13[[((i+17)%%28)+1]]
  kdeWeek2pointPredictionforILIReduced[[i]] <- temp14[[((i+17)%%28)+1]]
  kdeWeek3pointPredictionforILIReduced[[i]] <- temp15[[((i+17)%%28)+1]]
  kdeWeek4pointPredictionforILIReduced[[i]] <- temp16[[((i+17)%%28)+1]]
}

#  Now after refining the data files we need to
#  look at the predictions for one week, two weeks,
#  three weeks, and four weeks ahead and determine the shape of the
#  Kolmogorov-Smirnov test statistics plots.

sum(is.na(kdeWeek1Reduced[[1]]))
```

```
## [1] 0
```

```r
sum(is.na(kdeWeek1pointPredictionforILIReduced[[1]]))
```

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

```r
# Both sets of data files have length of 28 now.
# So there are two groups of 28 discrete distributions.

# The testStats variable has now been turned into a
# list of four vectors, each of which serves the function
# of the original testStats variable.
# This is done in order that we can generate four graphs
# corresponding to one week, two weeks, three weeks, and
# four weeks ahead.
```

```r
testStats <- vector("list", 4)

for (i in 1:4) {
  testStats[[i]] <- numeric(28)
}

for (i in 1:28) {
  testStats[[1]][i] <- max(abs(kcdeWeek1[[i]]$Value - kdeWeek1Reduced[[i]]$Value))
  testStats[[2]][i] <- max(abs(kcdeWeek2[[i]]$Value - kdeWeek2Reduced[[i]]$Value))
  testStats[[3]][i] <- max(abs(kcdeWeek3[[i]]$Value - kdeWeek3Reduced[[i]]$Value))
  testStats[[4]][i] <- max(abs(kcdeWeek4[[i]]$Value - kdeWeek4Reduced[[i]]$Value))
}




# We also want to show the true ILI data
# to assess a relationship between the
# models' predictions' differences and the ILI
# data for each week from
# https://gis.cdc.gov/grasp/fluview/fluportaldashboard.html
# You're going to need to download the ILINet data

# You're also going to need to delete the first row to help
# R load the file properly, as this is what I have done.

ILINet <- read.csv("C:/Users/gladi/Downloads/ILINet.csv")

# Taking into account that both of our models
# only predicted stuff from week 43 of 2017 to
# week 18 of 2018 (inclusive), we have to remove all other weeks
# from our ILINet.csv file in order that the points
# line up properly on our graphs since we are doing a
# visual comparison within a specific range.

# So we select the data from 2017 week 43 to
# 2018 week 18
weightedILIRange <- ILINet[4:31,]$X..WEIGHTED.ILI

trueILIPlot <- gf_point(weightedILIRange ~ seq_along(weightedILIRange),
                        xlab = "", ylab = "", title = "True % Weighted ILI Values",
                        col = "red") %>% gf_line(col = "red") +
  scale_y_continuous(breaks = c(1, 2, 3, 4, 5, 6, 7, 8)) +
  scale_x_continuous(breaks = c(1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27),
                     labels = c("1" = "w43", "3" = "w45", "5" = "w47", "7" = "w49",
                                "9" = "w51", "11" = "w1", "13" = "w3", "15" = "w5",
                                "17" = "w7", "19" = "w9", "21" = "w11", "23" = "w13",
                                "25" = "w15", "27" = "w17")) +
  theme(text = element_text(size = 9))

# We also define the weeks on the x-axis to
# avoid confusion.
# Because the units for our ILI plot
# are % ILI and the units for our other
# plots are K-S Statistics measured in
# probabilities assigned to each bin
# where each bin represents a range of
# % ILIs, our units are different so we will
# not overlay the graphs but will have to
# consider them separately.

ksStatsOneWeekAhead <- gf_point(testStats[[1]] ~ seq_along(testStats[[1]]),
                                xlab = "", ylab = "", title = "One Week Ahead",
                                col = "blue") %>%  gf_lims(y = c(0, 0.5)) %>%
  gf_line(col = "blue")+ scale_x_continuous(breaks = c(1, 3, 5, 7, 9, 11, 13,
                                                       15, 17, 19, 21, 23, 25,
                                                       27),
                        labels = c("1" = "w43", "3" = "w45", "5" = "w47",
                                   "7" = "w49", "9" = "w51", "11" = "w1",
                                   "13" = "w3", "15" = "w5", "17" = "w7",
                                   "19" = "w9", "21" = "w11", "23" = "w13",
                                   "25" = "w15", "27" = "w17")) +
  theme(text = element_text(size = 9))

ksStatsTwoWeeksAhead <- gf_point(testStats[[2]] ~ seq_along(testStats[[2]]),
                          xlab = "", ylab = "", title = "Two Weeks Ahead",
                          col = "blue") %>%  gf_lims(y = c(0, 0.5)) %>%
  gf_line(col = "blue") + scale_x_continuous(breaks = c(1, 3, 5, 7, 9, 11, 13,
                                                        15, 17, 19, 21, 23, 25, 27),
                labels = c("1" = "w43", "3" = "w45", "5" = "w47", "7" = "w49",
                           "9" = "w51", "11" = "w1", "13" = "w3", "15" = "w5",
                           "17" = "w7", "19" = "w9", "21" = "w11", "23" = "w13",
                           "25" = "w15", "27" = "w17")) +
  theme(text = element_text(size = 9))

ksStatsThreeWeekskAhead <- gf_point(testStats[[3]] ~ seq_along(testStats[[3]]),
                              xlab = "", ylab = "", title = "Three Weeks Ahead",
                              col = "blue") %>%  gf_lims(y = c(0, 0.5)) %>%
  gf_line(col = "blue") + scale_x_continuous(breaks = c(1, 3, 5, 7, 9, 11,
                                  13, 15, 17, 19, 21, 23, 25, 27),
              labels = c("1" = "w43", "3" = "w45", "5" = "w47", "7" = "w49",
                         "9" = "w51", "11" = "w1", "13" = "w3", "15" = "w5",
                         "17" = "w7", "19" = "w9", "21" = "w11", "23" = "w13",
                         "25" = "w15", "27" = "w17")) +
  theme(text = element_text(size = 9))

ksStatsFourWeeksAhead <- gf_point(testStats[[4]] ~ seq_along(testStats[[4]]),
                            xlab = "", ylab = "", title = "Four Weeks Ahead",
                            col = "blue") %>%  gf_lims(y = c(0, 0.5)) %>%
  gf_line(col = "blue") + scale_x_continuous(breaks = c(1, 3,
                          5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27),
            labels = c("1" = "w43", "3" = "w45", "5" = "w47", "7" = "w49",
                "9" = "w51", "11" = "w1", "13" = "w3", "15" = "w5", "17" = "w7",
          "19" = "w9", "21" = "w11", "23" = "w13", "25" = "w15", "27" = "w17")) +
  theme(text = element_text(size = 9))
```

```r
# Having stored these plots we will now
# look at both distributions and compare them.

# We know that our re-ordering of the files in the folder was
# probably successful because our following first two values of
# 0.217 and 0.255 are matched by the first graph (for one week
# ahead) - in the real-time-component models folder,
# kcdeFiles[[19]] corresponds to EW43-2017-ReichLab_kcde
# and kdeFiles[[23]] corresponds to EW43-2017-ReichLab_kde
# in the ReichLab_kcde and ReichLab_kde folders respectively.

tmp <- USNationalXWeeksAhead(kcdeFiles[[19]], "week one", "no")
tmp2 <- USNationalXWeeksAhead(kdeFiles[[23]], "week one", "no")
tmp3 <- USNationalXWeeksAhead(kcdeFiles[[20]], "week one", "no")
tmp4 <- USNationalXWeeksAhead(kdeFiles[[24]], "week one", "no")

max(abs(tmp$Value - tmp2$Value))
```

```
## [1] 0.2170192
```

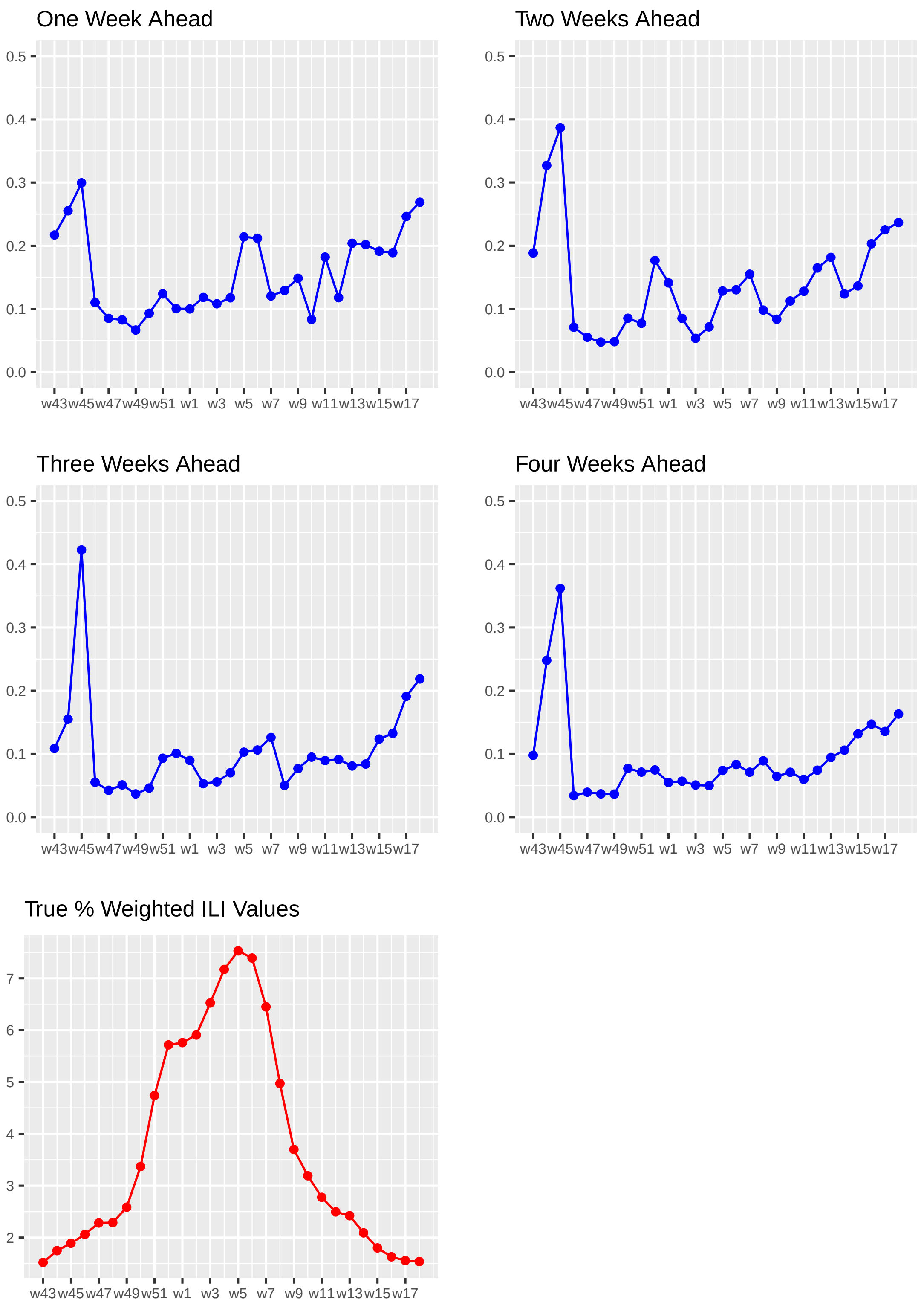
```r
max(abs(tmp3$Value - tmp4$Value))
```

```
## [1] 0.2552918
```

```r
# http://reichlab.io/assets/images/logo/nav-logo.png
img <- readPicture("C:/Users/gladi/Downloads/simple.svg")
imgGrob <- gTree(children=gList(pictureGrob(img)))
```

```r
plot <- grid.arrange(ksStatsOneWeekAhead, ksStatsTwoWeeksAhead,
             ksStatsThreeWeekskAhead, ksStatsFourWeeksAhead,
             trueILIPlot, imgGrob, top = paste("Comparing the Reich Lab's KCDE",
" and KDE Models  \n for Selected Weeks of 2017-2018 (Week 43, 2017 to ",
"Week 18, 2018) \n Kolmogorov-Smirnov Test Statistics between our KCDE and ",
"KDE Models' \n Assigned Probabilities to All ILI Bins (increment 0.1)", sep = ""))
```

Comparing the Reich Lab's KCDE and KDE Models
for Selected Weeks of 2017−2018 (Week 43, 2017 to Week 18, 2018)
Kolmogorov−Smirnov Test Statistics between our KCDE and KDE Models'
Assigned Probabilities to All ILI Bins (increment 0.1)



```r
# It looks like the maximum distance is much
# more for the first three starting weeks
# between these two models.

plot
```

```
## TableGrob (4 x 2) "arrange": 7 grobs
##   z     cells    name                grob
## 1 1 (2-2,1-1) arrange      gtable[layout]
## 2 2 (2-2,2-2) arrange      gtable[layout]
## 3 3 (3-3,1-1) arrange      gtable[layout]
## 4 4 (3-3,2-2) arrange      gtable[layout]
## 5 5 (4-4,1-1) arrange      gtable[layout]
## 6 6 (4-4,2-2) arrange gTree[GRID.gTree.15]
## 7 7 (1-1,1-2) arrange  text[GRID.text.248]
```

```r
typeof(plot)
```

```
## [1] "list"
```

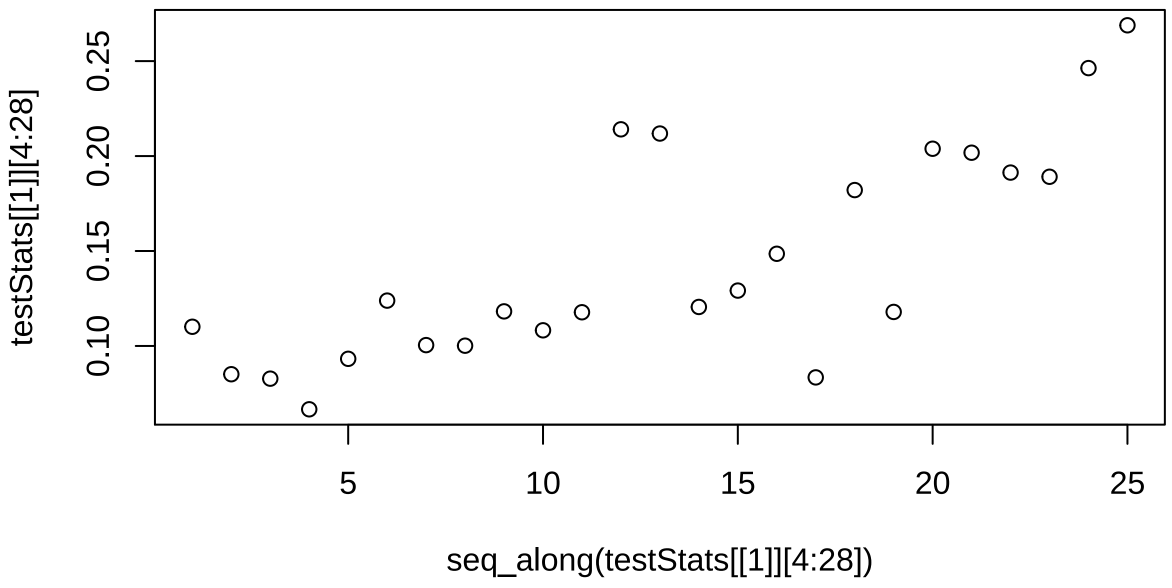
The test statistics seems to oscillate to some degree, and the initial spike in their values seems to indicate that a linear regression still is most likely not the best fit.

The true ILI of course takes on a positive value that both models try to predict but with fairly significant differences. P-Values should be extracted from these test statistics to determine the magnitude of this significance.

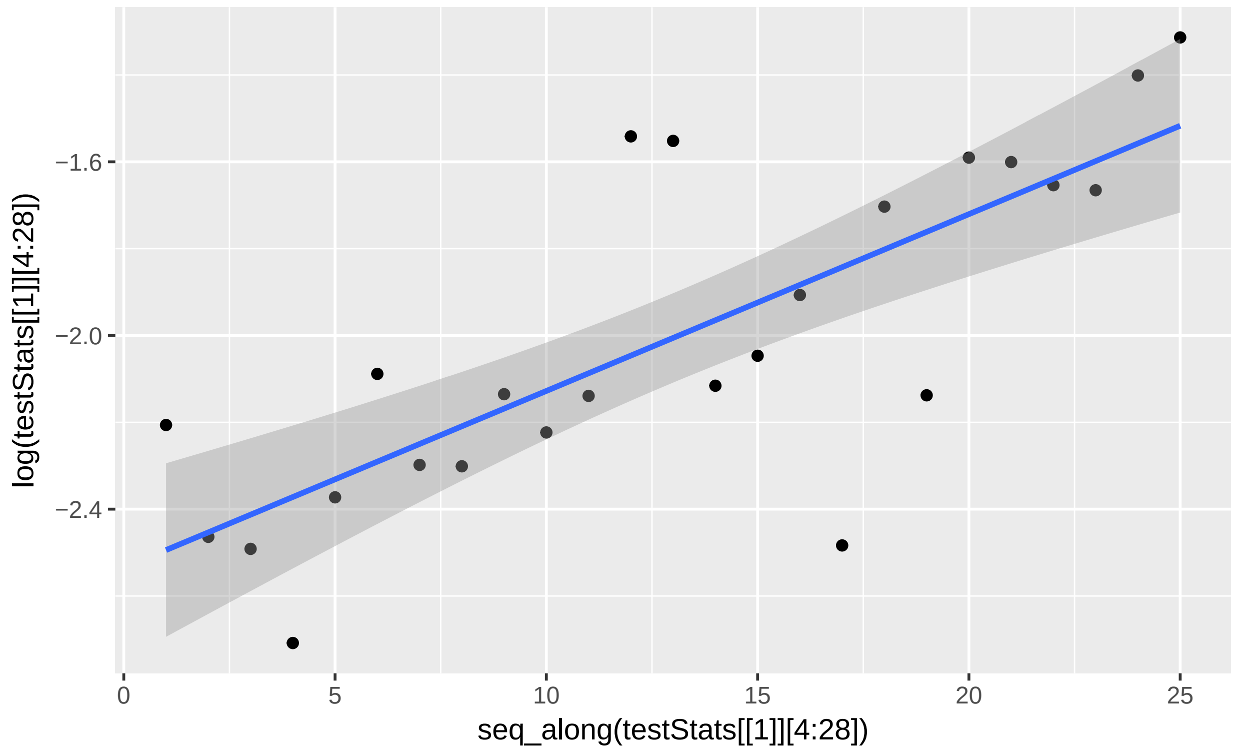
It makes sense that our models tend to differ more as they try to predict further into the future.

With some transformations it might be possible to obtain something resembling linearity:

```r
# Omitting the first three points,
# We are going to graph the test statistics
# for one week ahead.
plot(testStats[[1]][4:28] ~ seq_along(testStats[[1]][4:28]))
```



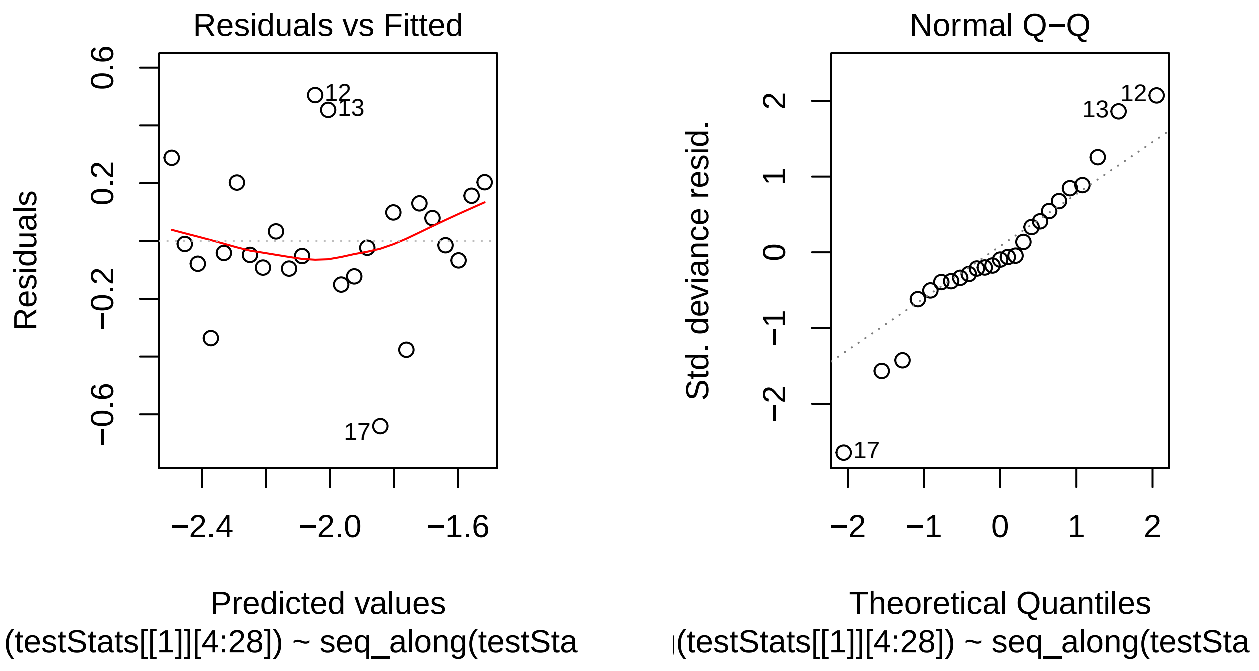
```r
# A log transformation might be necessary:
gf_point(log(testStats[[1]][4:28]) ~ seq_along(testStats[[1]][4:28])) +
  geom_smooth(method = 'lm', formula = y ~ x)
```

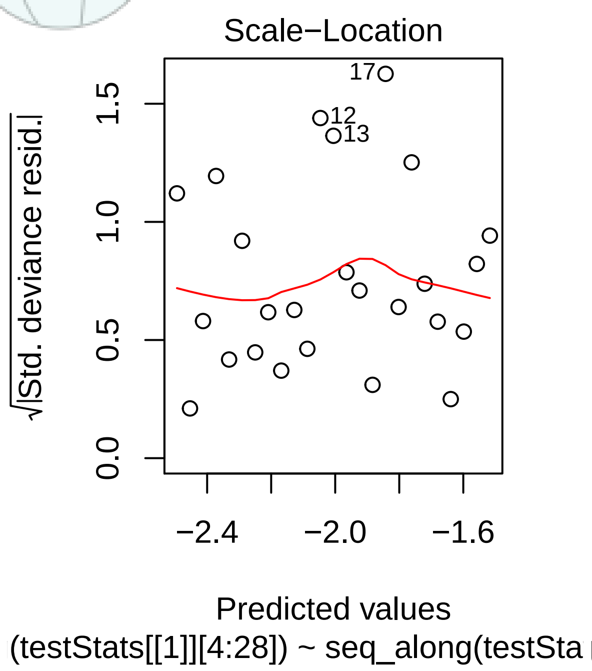


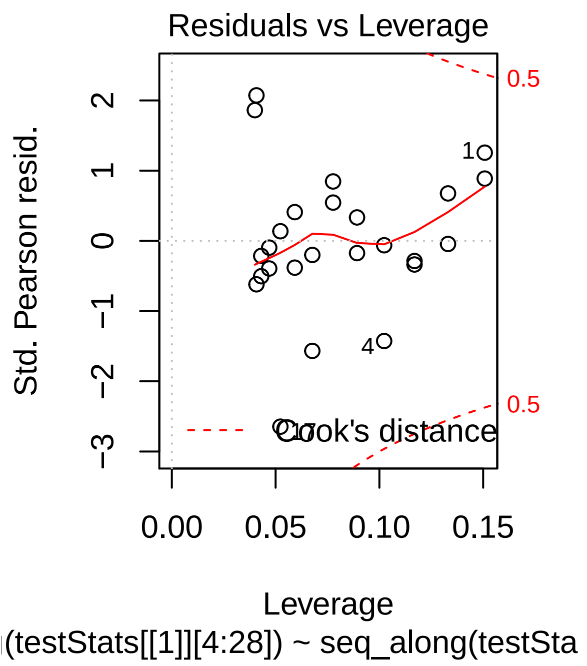
```r
lineOfBestFit <- glm((log(testStats[[1]][4:28]) ~ seq_along(testStats[[1]][4:28])))

stepAIC(lineOfBestFit)
```

```
## Start:  AIC=5.31
## log(testStats[[1]][4:28]) ~ seq_along(testStats[[1]][4:28])
##
##                                    Df Deviance     AIC
## <none>                                  1.4239  5.3095
## - seq_along(testStats[[1]][4:28])  1    3.5792 26.3534

##
## Call:  glm(formula = log(testStats[[1]][4:28]) ~ seq_along(testStats[[1]][4:28]))
##
## Coefficients:
##                      (Intercept)  seq_along(testStats[[1]][4:28])
##                         -2.53493                          0.04072
##
## Degrees of Freedom: 24 Total (i.e. Null);  23 Residual
## Null Deviance:       3.579
## Residual Deviance: 1.424     AIC: 5.31
```

```r
plot(lineOfBestFit)
```







```r
for (i in c(1,2)) { # We want to do a Wald Test
  print(1 - pchisq((coef(lineOfBestFit)[i] / sqrt(vcov(lineOfBestFit)[i,i]))^2, df = 1))
}
```

```
## (Intercept)
##           0
## seq_along(testStats[[1]][4:28])
##                     3.625581e-09
```

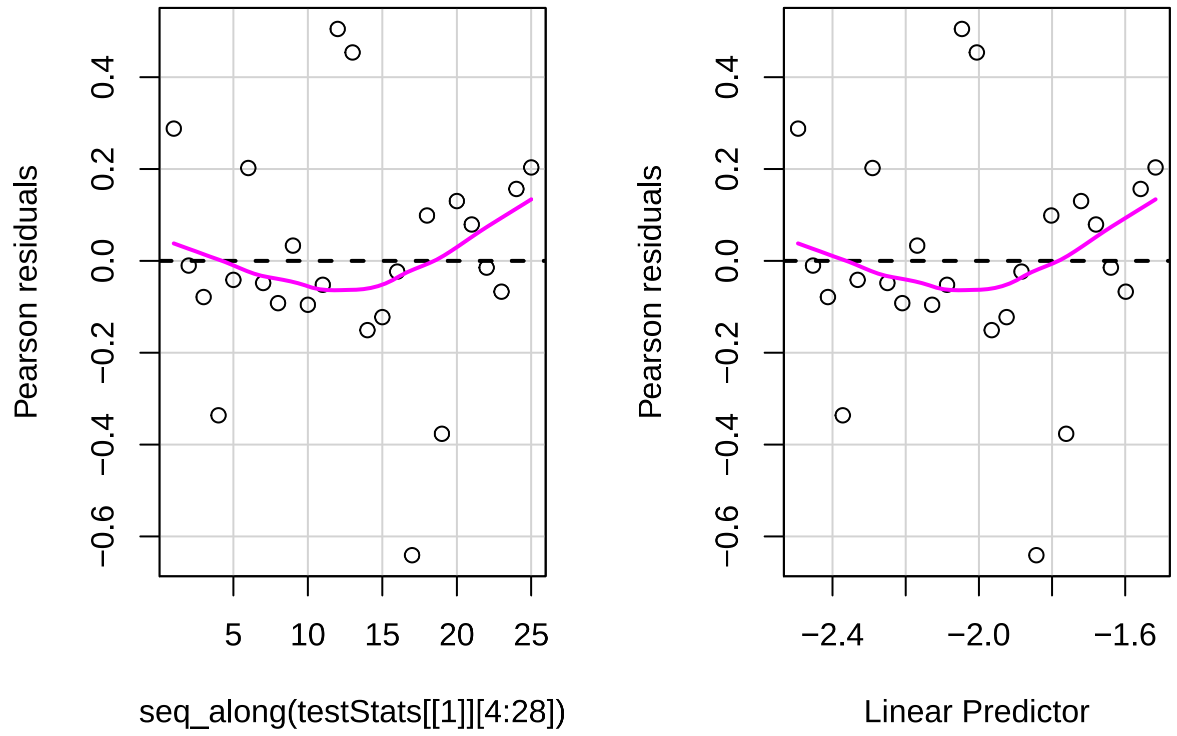
```r
anova(lineOfBestFit, stepAIC(lineOfBestFit), test = "Chisq")
```

```
## Start:  AIC=5.31
## log(testStats[[1]][4:28]) ~ seq_along(testStats[[1]][4:28])
##
##                                   Df Deviance     AIC
## <none>                                 1.4239  5.3095
## - seq_along(testStats[[1]][4:28])  1   3.5792 26.3534

## Analysis of Deviance Table
##
## Model 1: log(testStats[[1]][4:28]) ~ seq_along(testStats[[1]][4:28])
## Model 2: log(testStats[[1]][4:28]) ~ seq_along(testStats[[1]][4:28])
##   Resid. Df Resid. Dev Df Deviance Pr(>Chi)
## 1        23     1.4239
## 2        23     1.4239  0        0
```

```r
lineOfBestFit$residuals
```

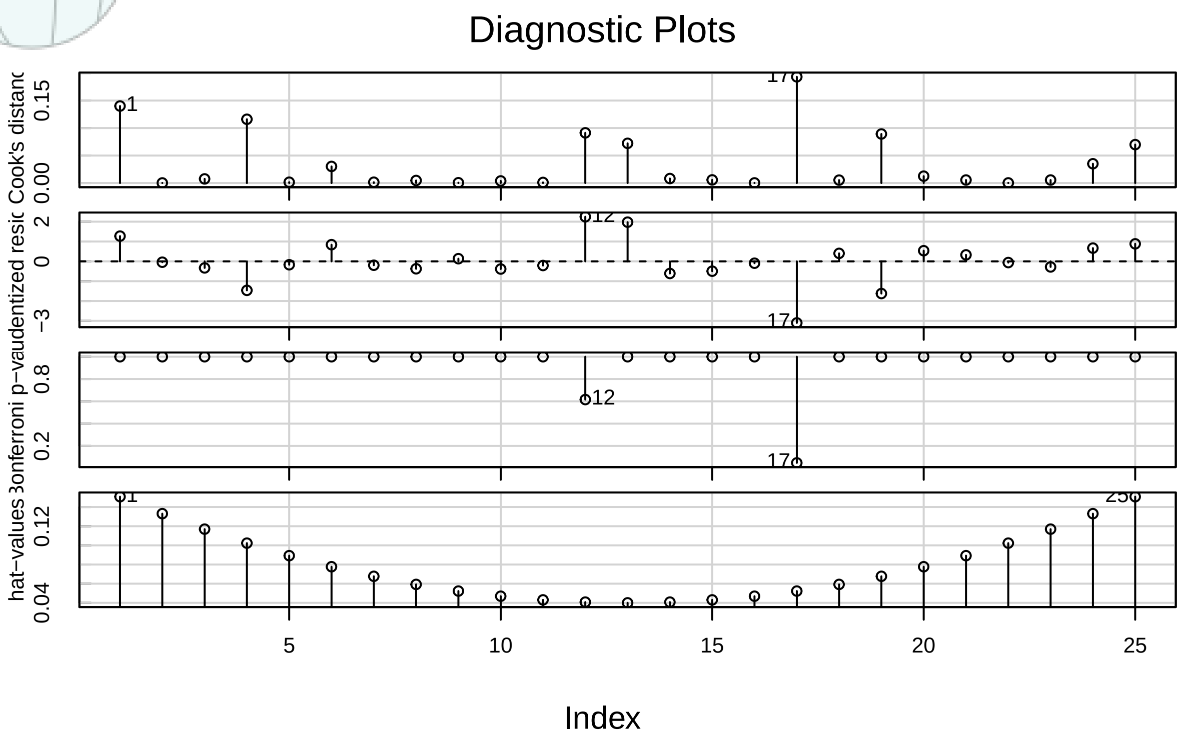
```
##           1           2           3           4           5           6
##  0.28798620 -0.01026554 -0.07873114 -0.33625642 -0.04138828  0.20219610
##           7           8           9          10          11          12
## -0.04814880 -0.09206213  0.03324661 -0.09557368 -0.05213136  0.50492935
##          13          14          15          16          17          18
##  0.45378654 -0.15082361 -0.12250014 -0.02338258 -0.64083688  0.09878744
##          19          20          21          22          23          24
## -0.37650423  0.13023001  0.07924362 -0.01472137 -0.06708226  0.15657827
##          25
##  0.20342428
```

```
residualPlots(lineOfBestFit, tests = TRUE)
```



```
##                                  Test stat Pr(>|Test stat|)
## seq_along(testStats[[1]][4:28])     0.0267           0.8702
```

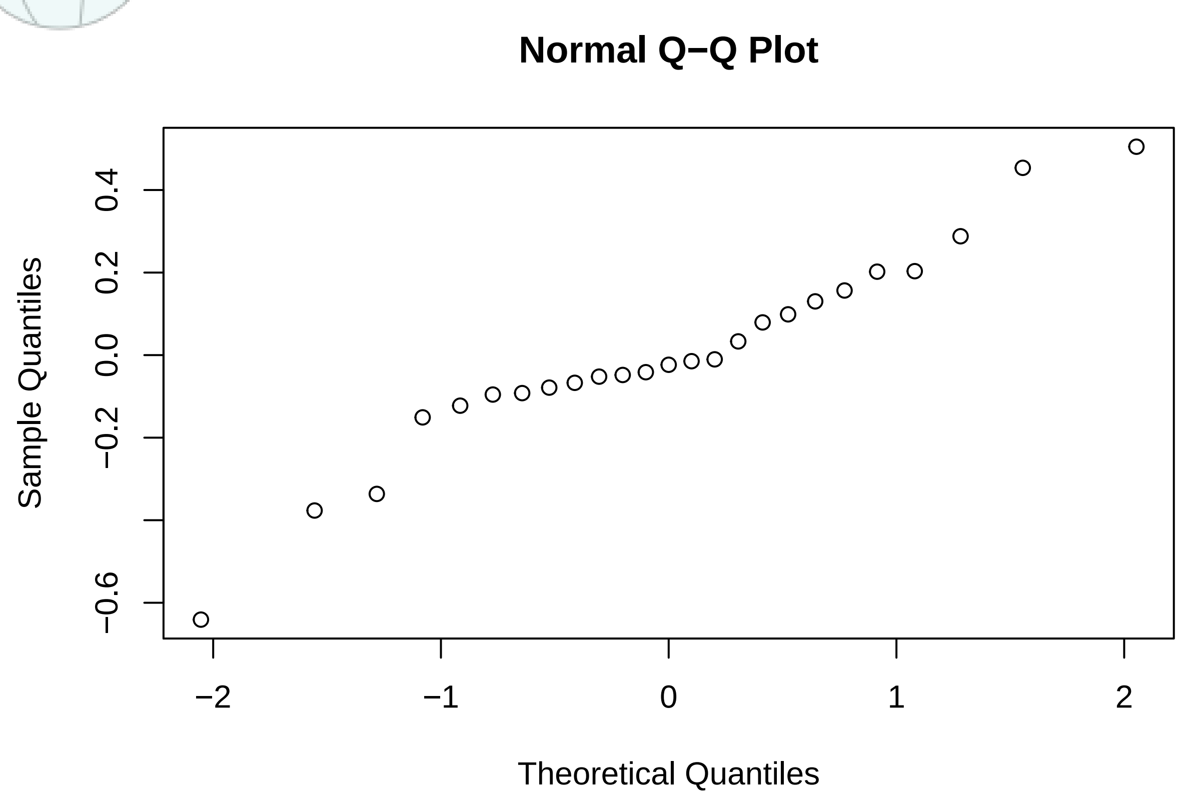
```
infIndexPlot(lineOfBestFit)
```



```r
summary(lineOfBestFit)
```

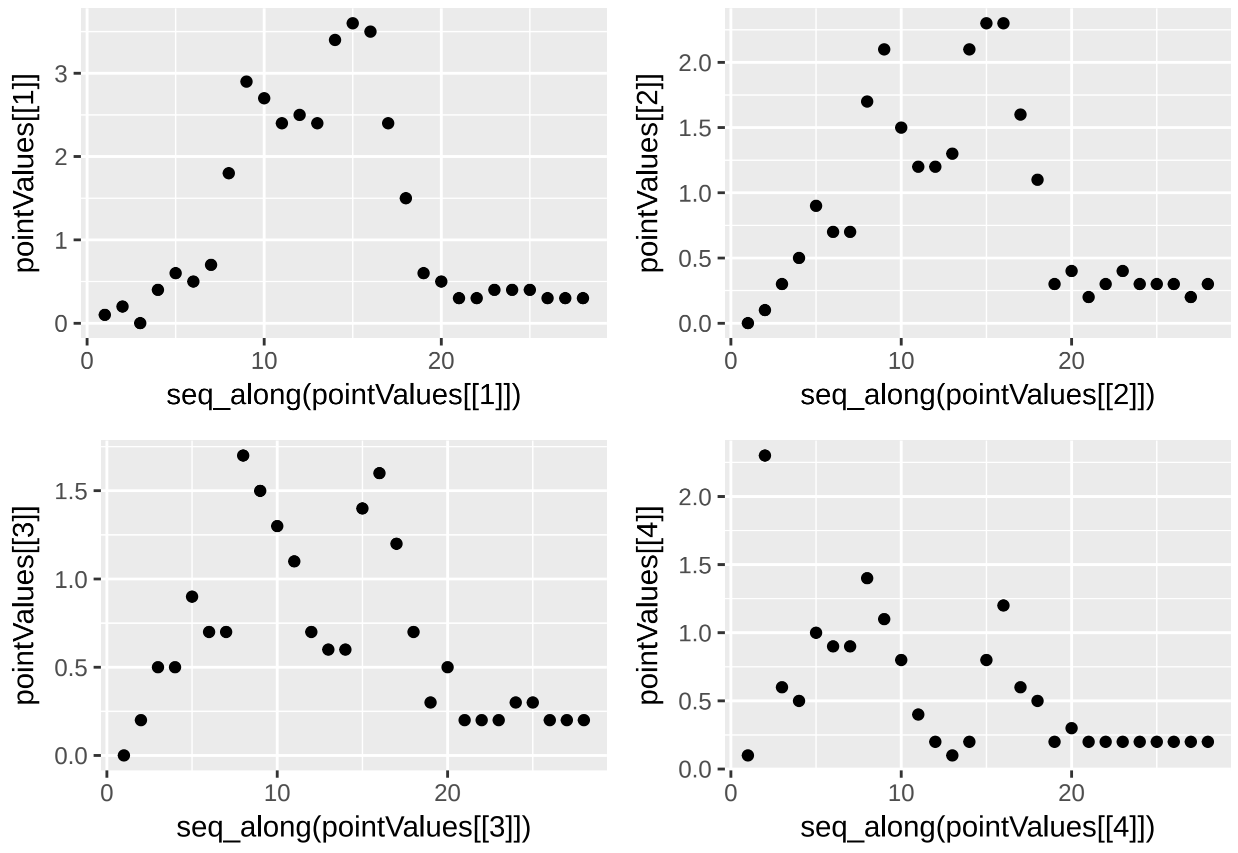
```
##
## Call:
## glm(formula = (log(testStats[[1]][4:28]) ~ seq_along(testStats[[1]][4:28])))
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -0.64084  -0.09206  -0.02338   0.13023   0.50493
##
## Coefficients:
##                                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)                    -2.534935   0.102588  -24.71  < 2e-16 ***
## seq_along(testStats[[1]][4:28])  0.040718   0.006901    5.90 5.15e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 0.06190752)
##
##     Null deviance: 3.5792  on 24  degrees of freedom
## Residual deviance: 1.4239  on 23  degrees of freedom
## AIC: 5.3095
##
## Number of Fisher Scoring iterations: 2
```

```
qqnorm(lineOfBestFit$residuals)
```



However, there is no definitive indication that the test statistics should come from a commonly known function such as an exponential one, so trying to find a linear function is probably not a high-yield path for most comparisons between models.

```
# Now, our second set of graphs shows the maximum
# distance between the point values for each week.

# This is different from the K-S test.

pointValues <- vector("list", 4)

for (i in 1:4) {
  pointValues[[i]] <- numeric(28)
} # Now that we have populated this list,

for (i in 1:28) {
  pointValues[[1]][i] <- max(abs(kcdeWeek1pointPredictionforILI[[i]]$Value -
                                   kdeWeek1pointPredictionforILIReduced[[i]]$Value))
  pointValues[[2]][i] <- max(abs(kcdeWeek2pointPredictionforILI[[i]]$Value -
                                   kdeWeek2pointPredictionforILIReduced[[i]]$Value))
  pointValues[[3]][i] <- max(abs(kcdeWeek3pointPredictionforILI[[i]]$Value -
                                   kdeWeek3pointPredictionforILIReduced[[i]]$Value))
  pointValues[[4]][i] <- max(abs(kcdeWeek4pointPredictionforILI[[i]]$Value -
                                   kdeWeek4pointPredictionforILIReduced[[i]]$Value))
}

pvDifferenceWeek1 <- gf_point(pointValues[[1]] ~ seq_along(pointValues[[1]]))
pvDifferenceWeek2 <- gf_point(pointValues[[2]] ~ seq_along(pointValues[[2]]))
pvDifferenceWeek3 <- gf_point(pointValues[[3]] ~ seq_along(pointValues[[3]]))
pvDifferenceWeek4 <- gf_point(pointValues[[4]] ~ seq_along(pointValues[[4]]))

grid.arrange(pvDifferenceWeek1, pvDifferenceWeek2, pvDifferenceWeek3, pvDifferenceWeek4)
```



Appendix:

line <- glm(testStats[[1]][4:28] ~ seq_along(testStats[[1]][4:28])) plot(allEffects(lineOfBestFit))