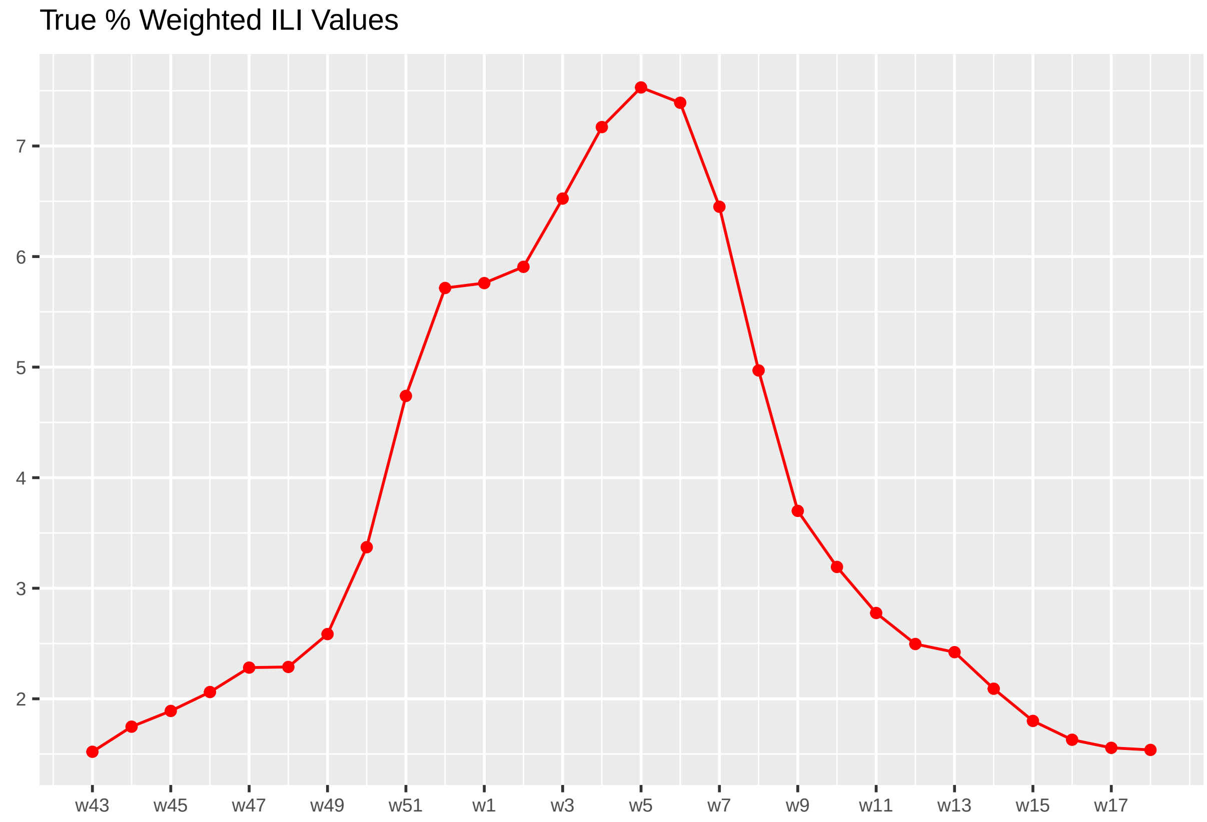
# Kolmogorov-Smirnov Test Regression, US National

*Dean Gladish and Nick Reich*

*July 11th, 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


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)

# 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.


# 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
```

```r
# 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))

trueILIPlot
```



```r
# 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.


# 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/",
                                  "ReichLab_kde", "/*.csv", sep = "")),
                   read.csv)


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



######################length(kdeWeek1)

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 o
```

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

# 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.

testStats <- vector("list", 4)

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

# In order to generate the CDF for the true K-S
# stats, first we're going to need to generate
# a vector corresponding to the actual CDF.
trueTestStats <- vector("list", 4)
for (i in 1:4) {
  trueTestStats[[i]] <- numeric(28)
}

kcdeWeek1CDF <- kcdeWeek1
kcdeWeek2CDF <- kcdeWeek2
kcdeWeek3CDF <- kcdeWeek3
```

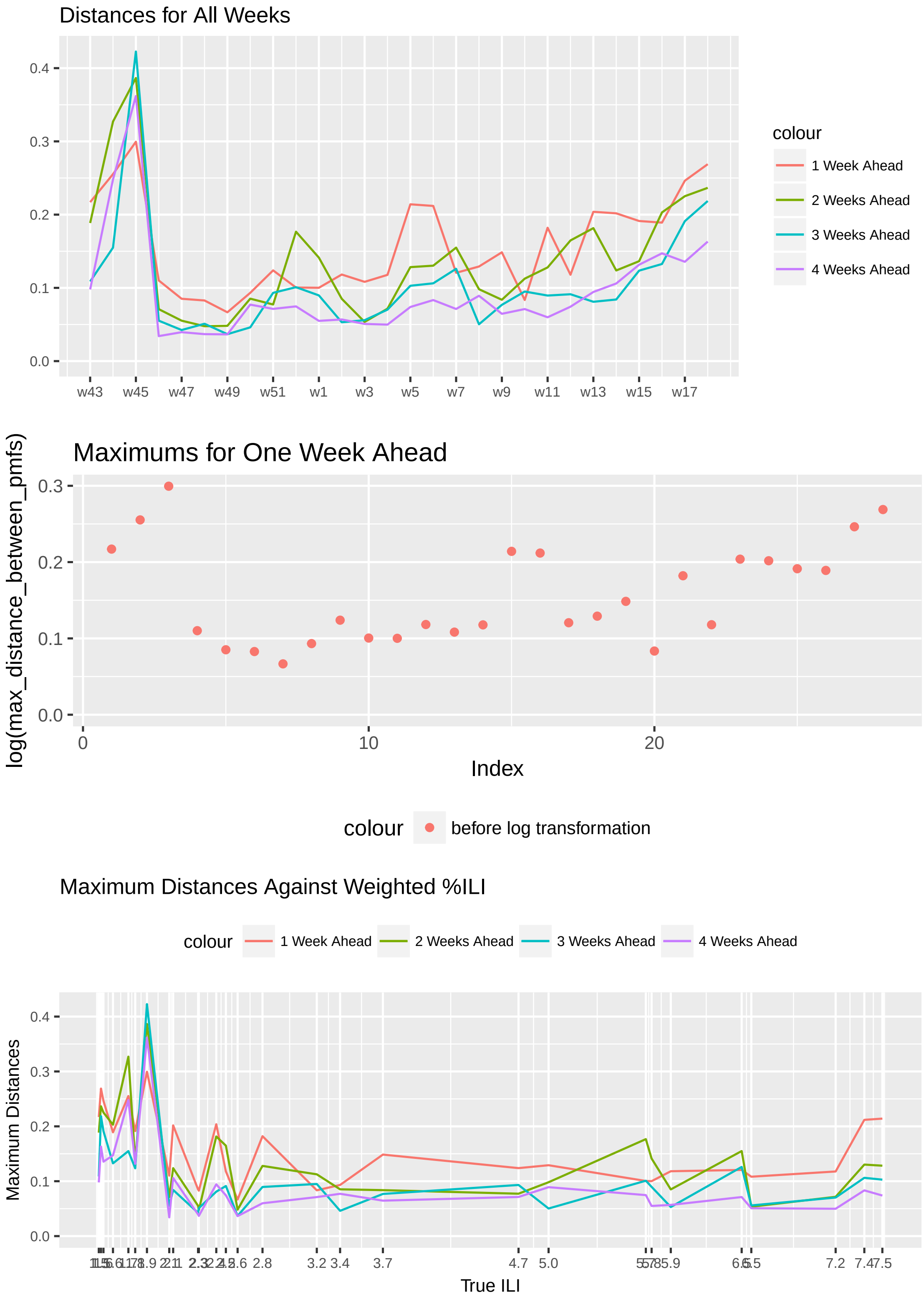
```
kcdeWeek4CDF <- kcdeWeek4

kdeWeek1CDF <- kdeWeek1Reduced
kdeWeek2CDF <- kdeWeek2Reduced
kdeWeek3CDF <- kdeWeek3Reduced
kdeWeek4CDF <- kdeWeek4Reduced

for (i in 1:28) {
  for (j in 1:131) {
    kcdeWeek1CDF[[i]]$Value[j] <- sum(kcdeWeek1[[i]]$Value[0:j])
    kcdeWeek2CDF[[i]]$Value[j] <- sum(kcdeWeek2[[i]]$Value[0:j])
    kcdeWeek3CDF[[i]]$Value[j] <- sum(kcdeWeek3[[i]]$Value[0:j])
    kcdeWeek4CDF[[i]]$Value[j] <- sum(kcdeWeek4[[i]]$Value[0:j])

    kdeWeek1CDF[[i]]$Value[j] <- sum(kdeWeek1Reduced[[i]]$Value[0:j])
    kdeWeek2CDF[[i]]$Value[j] <- sum(kdeWeek2Reduced[[i]]$Value[0:j])
    kdeWeek3CDF[[i]]$Value[j] <- sum(kdeWeek3Reduced[[i]]$Value[0:j])
    kdeWeek4CDF[[i]]$Value[j] <- sum(kdeWeek4Reduced[[i]]$Value[0:j])
  }
}


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))
}
for (i in 1:28) {
  trueTestStats[[1]][i] <- max(abs(kcdeWeek1CDF[[i]]$Value - kdeWeek1CDF[[i]]$Value))
  trueTestStats[[2]][i] <- max(abs(kcdeWeek2CDF[[i]]$Value - kdeWeek2CDF[[i]]$Value))
  trueTestStats[[3]][i] <- max(abs(kcdeWeek3CDF[[i]]$Value - kdeWeek3CDF[[i]]$Value))
  trueTestStats[[4]][i] <- max(abs(kcdeWeek4CDF[[i]]$Value - kdeWeek4CDF[[i]]$Value))
}








# I also want to put the 5% significance threshold
# on the graph.

######################length(kcdeWeek1[[1]]$Value)

# This is 131, so we are basing each
# test statistic on a data set of size 131.

# Basically our thresholds should be

######################(1.62762)/(sqrt(131))
```

```
######################(1.3581)/(sqrt(131))

######################(1.22385)/(sqrt(131))


# based on this website http://www.real-statistics.com
# /statistics-tables/kolmogorov-smirnov-table/
# for significance levels of 0.1, 0.05, and 0.01.
# These are

m <- max(testStats[[1]], testStats[[2]], testStats[[3]], testStats[[4]])

ksStatsAllWeeksAhead <- gf_line(testStats[[1]] ~ seq_along(testStats[[1]]),
                                xlab = "", ylab = "", title = "Distances for All Weeks",
                                color = ~"1 Week Ahead") %>%  gf_lims(y = c(0, m)) %>%
  gf_line(testStats[[2]] ~ seq_along(testStats[[2]]),
                                xlab = "", ylab = "", title = "Distances for All Weeks",
                                color = ~"2 Weeks Ahead") %>%  gf_lims(y = c(0, m)) %>%
gf_line(testStats[[3]] ~ seq_along(testStats[[3]]),
                                xlab = "", ylab = "", title = "Distances for All Weeks",
                                color = ~"3 Weeks Ahead") %>%  gf_lims(y = c(0, m)) %>%
gf_line(testStats[[4]] ~ seq_along(testStats[[4]]),
                                xlab = "", ylab = "", title = "Distances for All Weeks",
                                color = ~"4 Weeks Ahead") %>%  gf_lims(y = c(0, m)) + scale_x_continuou
                          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))


# 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))
######################max(abs(tmp3$Value - tmp4$Value))

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

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

# We're also going to try to see if the statistics
# are linear.
# We're going to include all points and graph the maximums for one week ahead.
# The gray dots are points before the log transformation.

# We are going to graph the maximums
# for one week ahead.  A log transformat-
# ion might be needed so we're going to use that
# and include the original points in orange.

linearityTestMaximums <- gf_point(log(testStats[[1]][1:28]) ~ seq_along(testStats[[1]][1:28]),
         title = "Maximums for One Week Ahead",
         xlab = "Index", ylab = "log(max_distance_between_pmfs)") %>% gf_lims(y = c(0, max(testStats[[1
  gf_point(testStats[[1]][1:28] ~ seq_along(testStats[[1]][1:28]),
           xlab = "Index",
           color = ~"before log transformation") %>% gf_lims(y = c(0, max(testStats[[1]]))) %>%
  gf_theme(legend.position = "bottom") +
  geom_smooth(method = 'lm', formula = y ~ x)


lineOfBestFit <- glm((log(testStats[[1]][1:28]) ~ seq_along(testStats[[1]][1:28])))
######################stepAIC(lineOfBestFit)





actualILIagainstMaximums <-
  gf_line(testStats[[1]] ~ weightedILIRange,
           xlab = "True ILI", ylab = "Maximum Distances", title = "Maximum Distances Against Weighted %
           color = ~"1 Week Ahead") %>% gf_theme(legend.position = "top") %>% gf_lims(y = c(0, m)) %>%

  gf_line(testStats[[2]] ~ weightedILIRange,
           xlab = "", ylab = "", title = "Maximum Distances Against Weighted %ILI",
           color = ~"2 Weeks Ahead") %>% gf_lims(y = c(0, m)) %>%
  gf_line(testStats[[3]] ~ weightedILIRange,
           xlab = "", ylab = "", title = "Maximum Distances Against Weighted %ILI",
           color = ~"3 Weeks Ahead") %>% gf_lims(y = c(0, m)) %>%
  gf_line(testStats[[4]] ~ weightedILIRange,
           xlab = "True ILI", ylab = "Maximum Distances", title = "Maximum Distances Against Weighted %
           color = ~"4 Weeks Ahead") %>%
  gf_lims(y = c(0, m)) + scale_x_continuous(breaks = c(1.52071, 1.53749, 1.55644, 1.62950, 1.74765, 1.7
  theme(text = element_text(size = 9))
```

```
plot <- grid.arrange(ksStatsAllWeeksAhead,
             linearityTestMaximums, actualILIagainstMaximums, top = paste("Comparing the Reich Lab's KC
" and KDE Models  \n for Selected Weeks of 2017-2018 (Week 43, 2017 to ",
"Week 18, 2018) \n Maximum Differences between our KCDE and KDE Models' \n",
"Probability Mass Functions for 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)
Maximum Differences between our KCDE and KDE Models'
Probability Mass Functions for All ILI Bins (increment 0.1)

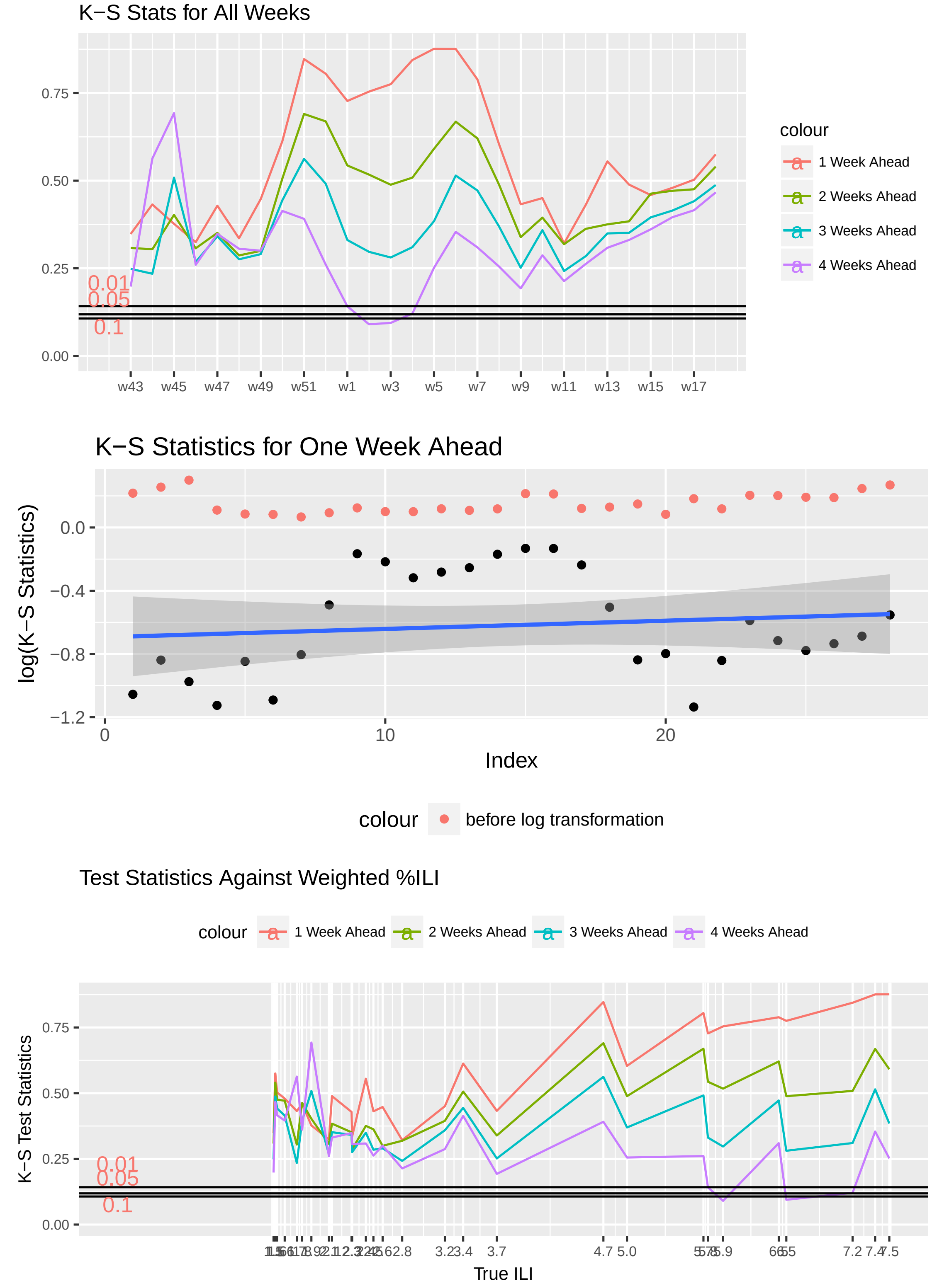


```
# It looks like the maximum distance is much
# more for the first three starting weeks
# between these two models.

######################plot
######################typeof(plot)

# By the same method we're going to derive the
# true, standard K-S test statistics from the
# cumulative distribution functions.

m2 <- max(trueTestStats[[1]], trueTestStats[[2]], trueTestStats[[3]], trueTestStats[[4]])

trueKSStatsAllWeeksAhead <- gf_line(trueTestStats[[1]] ~ seq_along(trueTestStats[[1]]),
                                xlab = "", ylab = "", title = "K-S Stats for All Weeks",
                                color = ~"1 Week Ahead") %>% gf_lims(y = c(0, m2)) %>%
  gf_line(trueTestStats[[2]] ~ seq_along(trueTestStats[[2]]),
                                xlab = "", ylab = "", title = "K-S Stats for All Weeks",
                                color = ~"2 Weeks Ahead") %>%  gf_lims(y = c(0, m2)) %>%
gf_line(trueTestStats[[3]] ~ seq_along(trueTestStats[[3]]),
                                xlab = "", ylab = "", title = "K-S Stats for All Weeks",
                                color = ~"3 Weeks Ahead") %>%  gf_lims(y = c(0, m2)) %>%
gf_line(trueTestStats[[4]] ~ seq_along(trueTestStats[[4]]),
                                xlab = "", ylab = "", title = "K-S Stats for All Weeks",
                                color = ~"4 Weeks Ahead") %>%  gf_lims(y = c(0, m2)) + scale_x_continuou
                          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)) + geom_hline(aes(yintercept = 0.1422058)) + geom_text(aes(0,0.142



# We are going to graph the K-S test
# statistics
# for one week ahead.  A log transformat-
# ion might be needed so we're going to use that
# and include the original points in orange.

linearityTestKSStatistics <- gf_point(log(trueTestStats[[1]][1:28]) ~ seq_along(trueTestStats[[1]][1:28]
         title = "K-S Statistics for One Week Ahead",
         xlab = "Index", ylab = "log(K-S Statistics)") %>%
  gf_point(testStats[[1]][1:28] ~ seq_along(testStats[[1]][1:28]),
           xlab = "Index",
           color = ~"before log transformation") %>% gf_theme(legend.position = "bottom") +
  geom_smooth(method = 'lm', formula = y ~ x)


lineOfBestFit2 <- glm((log(trueTestStats[[1]][1:28]) ~ seq_along(trueTestStats[[1]][1:28])))
######################stepAIC(lineOfBestFit2)


actualILIagainstKSStatistics <-
```

```
  gf_line(trueTestStats[[1]] ~ weightedILIRange,
           xlab = "True ILI", ylab = "K-S Test Statistics", title = "Test Statistics Against Weighted %
           color = ~"1 Week Ahead") %>% gf_theme(legend.position = "top") %>% gf_lims(y = c(0, m2)) %>%

  gf_line(trueTestStats[[2]] ~ weightedILIRange,
           xlab = "", ylab = "", title = "Test Statistics Against Weighted %ILI",
           color = ~"2 Weeks Ahead") %>% gf_lims(y = c(0, m2)) %>%
  gf_line(trueTestStats[[3]] ~ weightedILIRange,
           xlab = "", ylab = "", title = "Test Statistics Against Weighted %ILI",
           color = ~"3 Weeks Ahead") %>% gf_lims(y = c(0, m2)) %>%
  gf_line(trueTestStats[[4]] ~ weightedILIRange,
           xlab = "True ILI", ylab = "K-S Test Statistics", title = "Test Statistics Against Weighted %
           color = ~"4 Weeks Ahead") %>%
  gf_lims(y = c(0, m2)) + scale_x_continuous(breaks = c(1.52071, 1.53749, 1.55644, 1.62950, 1.74765, 1.
  theme(text = element_text(size = 9))+ geom_hline(aes(yintercept = 0.1422058)) + geom_text(aes(0,0.142
```

```
grid.arrange(trueKSStatsAllWeeksAhead, linearityTestKSStatistics, actualILIagainstKSStatistics, top = p
" 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 Empirical CDF for 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'
Empirical CDF for All ILI Bins (increment 0.1)

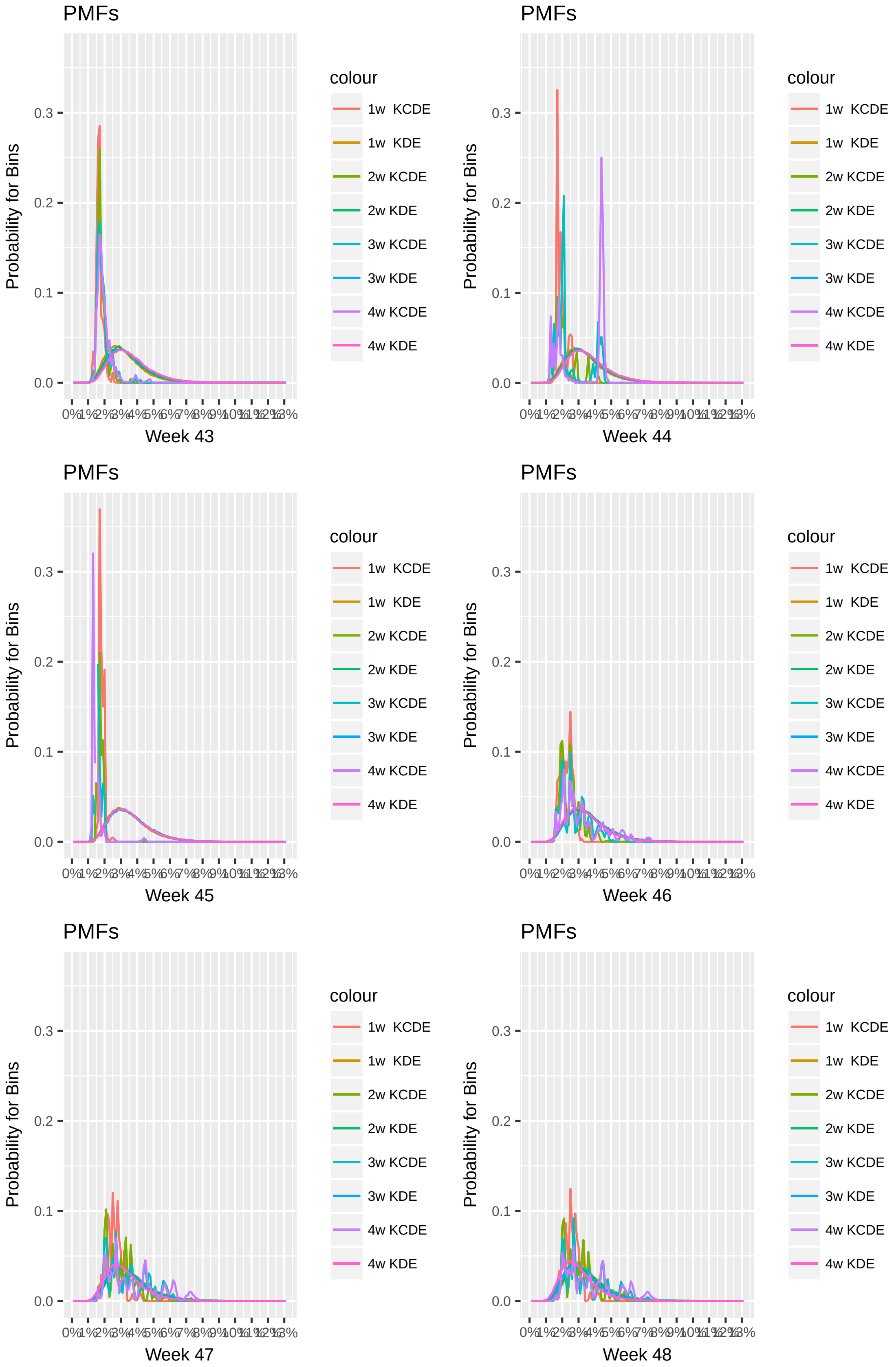


```
#The test statistics seems to oscillate to some degree, and the initial spike in their values seems to

#The true ILI of course takes on a positive value that both models try to predict but with fairly signi

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

#Before we look for possible linearity we should create graphs of each individual distribution in order
```

```
# The following code will generate graphs of the probabilities
# assigned to each bin for one week ahead, two weeks ahead,
# and three weeks ahead.
kcde1week <- kde1week <- kcde2weeks <- kde2weeks <-
  kcde3weeks <- kde3weeks <- kcde4weeks <- kde4weeks <-
  vector("list", 6)
for (i in 1:6) {
  kcde1week[[i]] <- kde1week[[i]] <- kcde2weeks[[i]] <-
    kde2weeks[[i]] <- kcde3weeks[[i]] <- kde3weeks[[i]] <-
    kcde4weeks[[i]] <- kde4weeks[[i]] <-
    numeric(131)
}

# Because kcdeweek1 is a data frame of size 131
# and we basically want to explain the fact that
# the first three K-S test statistics between the
# kcde and kde models are much larger than the rest,
# we use the first for-loop to populate the kcde1week
# list with four different vectors that contain all
# probability predictions for each bin for that week ahead.

# We arbitrarily chose to have four vectors in each
# list because this would show the probability
# distributions that correspond to the first
# four K-S test statistics on the graph and might
# provide some insight into which model is the
# culprit in the observed increased difference.

# Creating new vectors isn't actually necessary but will
# make the code for the actual graphs slightly smaller,
# which is what we want.  It will just allow us to focus
# on what we want.
for (j in 1:6) {
   for (i in 1:131) {
    kcde1week[[j]][[i]] <- kcdeWeek1[[j]]$Value[[i]]
    kde1week[[j]][[i]] <- kdeWeek1[[j]]$Value[[i]]
    kcde2weeks[[j]][[i]] <- kcdeWeek2[[j]]$Value[[i]]
    kde2weeks[[j]][[i]] <- kdeWeek2[[j]]$Value[[i]]
    kcde3weeks[[j]][[i]] <- kcdeWeek3[[j]]$Value[[i]]
    kde3weeks[[j]][[i]] <- kdeWeek3[[j]]$Value[[i]]
    kcde4weeks[[j]][[i]] <- kcdeWeek4[[j]]$Value[[i]]
    kde4weeks[[j]][[i]] <- kdeWeek4[[j]]$Value[[i]]
  }
}
```

```r
# Now we must also define the plots.
max1 <- max(kcde1week[[1]], kcde1week[[2]], kcde1week[[3]], kcde1week[[4]], kcde1week[[5]], kcde1week[[


plotw43AllWeeksAhead <- gf_line(kcde1week[[1]] ~ seq_along(kcde1week[[1]]), color = ~"1w  KCDE", xlab =
  gf_line(kcde2weeks[[1]] ~ seq_along(kcde2weeks[[1]]), color = ~"2w KCDE", xlab = "Week 43", ylab = ""
  gf_line(kde2weeks[[1]] ~ seq_along(kde2weeks[[1]]), color = ~"2w KDE") %>% gf_lims(y = c(0, max1))  %
  gf_line(kcde3weeks[[1]] ~ seq_along(kcde3weeks[[1]]), color = ~"3w KCDE", xlab = "Week 43", ylab = ""
  gf_line(kde3weeks[[1]] ~ seq_along(kde3weeks[[1]]), color = ~"3w KDE") %>% gf_lims(y = c(0, max1))  %
  gf_line(kcde4weeks[[1]] ~ seq_along(kcde4weeks[[1]]), color = ~"4w KCDE", xlab = "Week 43", ylab = "P
  gf_line(kde4weeks[[1]] ~ seq_along(kde4weeks[[1]]), color = ~"4w KDE") %>% gf_lims(y = c(0, max1)) +

plotw44AllWeeksAhead <- gf_line(kcde1week[[2]] ~ seq_along(kcde1week[[2]]), color = ~"1w  KCDE", xlab =
  gf_line(kcde2weeks[[2]] ~ seq_along(kcde2weeks[[2]]), color = ~"2w KCDE", xlab = "Week 44", ylab = ""
  gf_line(kde2weeks[[2]] ~ seq_along(kde2weeks[[2]]), color = ~"2w KDE") %>% gf_lims(y = c(0, max1))  %
  gf_line(kcde3weeks[[2]] ~ seq_along(kcde3weeks[[2]]), color = ~"3w KCDE", xlab = "Week 44", ylab = ""
  gf_line(kde3weeks[[2]] ~ seq_along(kde3weeks[[2]]), color = ~"3w KDE") %>% gf_lims(y = c(0, max1))  %
  gf_line(kcde4weeks[[2]] ~ seq_along(kcde4weeks[[2]]), color = ~"4w KCDE", xlab = "Week 44", ylab = "P
  gf_line(kde4weeks[[2]] ~ seq_along(kde4weeks[[2]]), color = ~"4w KDE") %>% gf_lims(y = c(0, max1)) +

plotw45AllWeeksAhead <- gf_line(kcde1week[[3]] ~ seq_along(kcde1week[[3]]), color = ~"1w  KCDE", xlab =
  gf_line(kcde2weeks[[3]] ~ seq_along(kcde2weeks[[3]]), color = ~"2w KCDE", xlab = "Week 45", ylab = ""
  gf_line(kde2weeks[[3]] ~ seq_along(kde2weeks[[3]]), color = ~"2w KDE") %>% gf_lims(y = c(0, max1))  %
  gf_line(kcde3weeks[[3]] ~ seq_along(kcde3weeks[[3]]), color = ~"3w KCDE", xlab = "Week 45", ylab = ""
  gf_line(kde3weeks[[3]] ~ seq_along(kde3weeks[[3]]), color = ~"3w KDE") %>% gf_lims(y = c(0, max1))  %
  gf_line(kcde4weeks[[3]] ~ seq_along(kcde4weeks[[3]]), color = ~"4w KCDE", xlab = "Week 45", ylab = "P
  gf_line(kde4weeks[[3]] ~ seq_along(kde4weeks[[3]]), color = ~"4w KDE") %>% gf_lims(y = c(0, max1)) +

plotw46AllWeeksAhead <- gf_line(kcde1week[[4]] ~ seq_along(kcde1week[[4]]), color = ~"1w  KCDE", xlab =
  gf_line(kcde2weeks[[4]] ~ seq_along(kcde2weeks[[4]]), color = ~"2w KCDE", xlab = "Week 46", ylab = ""
  gf_line(kde2weeks[[4]] ~ seq_along(kde2weeks[[4]]), color = ~"2w KDE") %>% gf_lims(y = c(0, max1))  %
  gf_line(kcde3weeks[[4]] ~ seq_along(kcde3weeks[[4]]), color = ~"3w KCDE", xlab = "Week 46", ylab = ""
  gf_line(kde3weeks[[4]] ~ seq_along(kde3weeks[[4]]), color = ~"3w KDE") %>% gf_lims(y = c(0, max1))  %
  gf_line(kcde4weeks[[4]] ~ seq_along(kcde4weeks[[4]]), color = ~"4w KCDE", xlab = "Week 46", ylab = "P
  gf_line(kde4weeks[[4]] ~ seq_along(kde4weeks[[4]]), color = ~"4w KDE") %>% gf_lims(y = c(0, max1)) +

plotw47AllWeeksAhead <- gf_line(kcde1week[[5]] ~ seq_along(kcde1week[[5]]), color = ~"1w  KCDE", xlab =
  gf_line(kcde2weeks[[5]] ~ seq_along(kcde2weeks[[5]]), color = ~"2w KCDE", xlab = "Week 47", ylab = ""
  gf_line(kde2weeks[[5]] ~ seq_along(kde2weeks[[5]]), color = ~"2w KDE") %>% gf_lims(y = c(0, max1))  %
  gf_line(kcde3weeks[[5]] ~ seq_along(kcde3weeks[[5]]), color = ~"3w KCDE", xlab = "Week 47", ylab = ""
  gf_line(kde3weeks[[5]] ~ seq_along(kde3weeks[[5]]), color = ~"3w KDE") %>% gf_lims(y = c(0, max1))  %
  gf_line(kcde4weeks[[5]] ~ seq_along(kcde4weeks[[5]]), color = ~"4w KCDE", xlab = "Week 47", ylab = "P
  gf_line(kde4weeks[[5]] ~ seq_along(kde4weeks[[5]]), color = ~"4w KDE") %>% gf_lims(y = c(0, max1)) +

plotw48AllWeeksAhead <- gf_line(kcde1week[[6]] ~ seq_along(kcde1week[[6]]), color = ~"1w  KCDE", xlab =
  gf_line(kcde2weeks[[6]] ~ seq_along(kcde2weeks[[6]]), color = ~"2w KCDE", xlab = "Week 48", ylab = ""
  gf_line(kde2weeks[[6]] ~ seq_along(kde2weeks[[6]]), color = ~"2w KDE") %>% gf_lims(y = c(0, max1))  %
  gf_line(kcde3weeks[[6]] ~ seq_along(kcde3weeks[[6]]), color = ~"3w KCDE", xlab = "Week 48", ylab = ""
  gf_line(kde3weeks[[6]] ~ seq_along(kde3weeks[[6]]), color = ~"3w KDE") %>% gf_lims(y = c(0, max1))  %
  gf_line(kcde4weeks[[6]] ~ seq_along(kcde4weeks[[6]]), color = ~"4w KCDE", xlab = "Week 48", ylab = "P
  gf_line(kde4weeks[[6]] ~ seq_along(kde4weeks[[6]]), color = ~"4w KDE") %>% gf_lims(y = c(0, max1)) +
# These allow us to see that the KCDE model is primarily the one that is
# responsible for these extreme deviations in K-S test statistics.
# As we can see, the KDE model largely remains the same as far as its
# probabilities for each bin are presented across each week (43, 44, 45,
# 46, 47, 48)

# We can also plot the actual ILI against the K-S
# test statistics to find a correlation.
# The correspondence of each ILI - K-S Test statistic
# pair is preserved in these graphs.

# In order to appropriately label the graph we're going to
# have to look at the values for true weighted ILI in their
# proper order.

######################sort(weightedILIRange, decreasing = F)
######################round(c(1.52071, 1.53749, 1.55644, 1.62950, 1.74765, 1.79983, 1.88999, 2.06099, 2


grid.arrange(plotw43AllWeeksAhead, plotw44AllWeeksAhead, plotw45AllWeeksAhead, plotw46AllWeeksAhead, plo
```

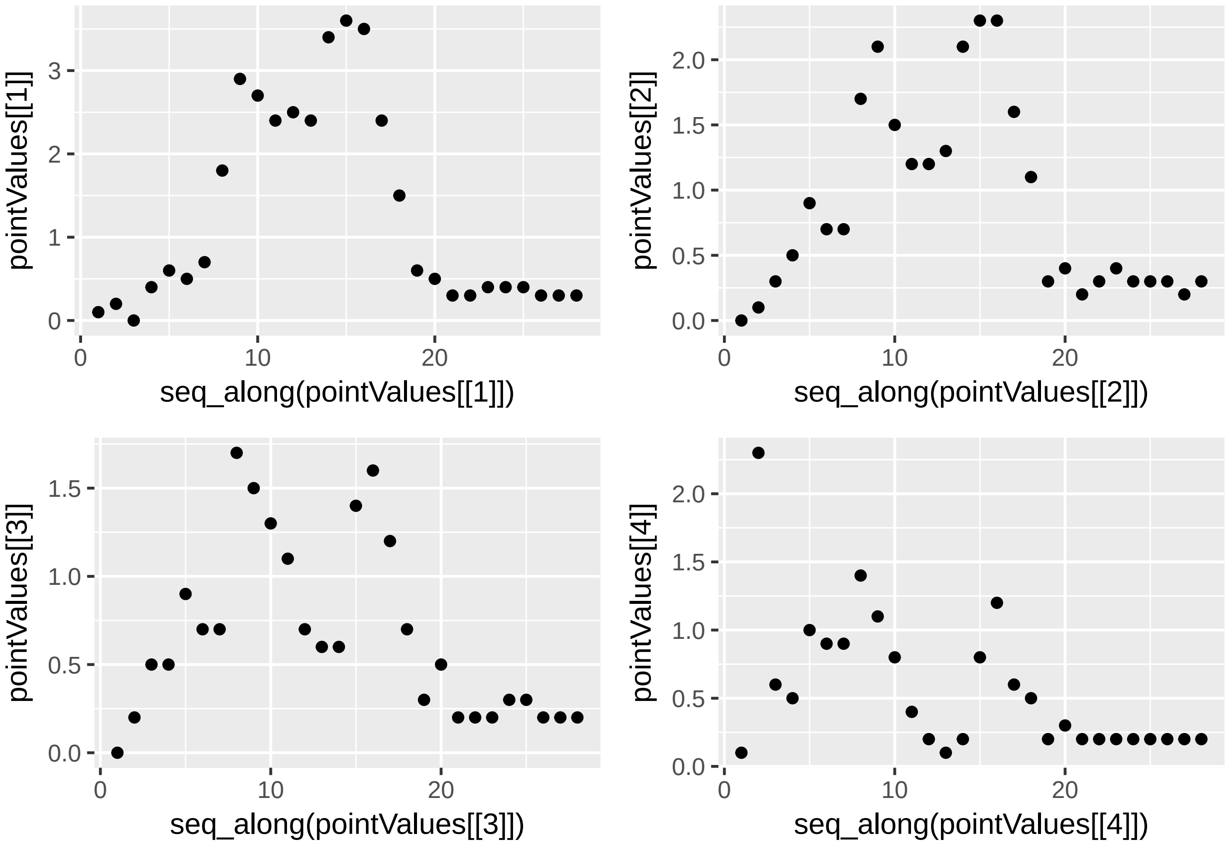
KCDE and KDE PMFs for %ILI, All Weeks Ahead



```
######################plot(lineOfBestFit)
```

```
######################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, di
######################}
######################anova(lineOfBestFit, stepAIC(lineOfBestFit), test = "Chisq")
######################lineOfBestFit$residuals
######################residualPlots(lineOfBestFit, tests = TRUE)
######################infIndexPlot(lineOfBestFit)
######################summary(lineOfBestFit)
######################qqnorm(lineOfBestFit$residuals)


#However, there is no definitive indication that the test statistics should come from a commonly known
```

```
# 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))
```