An Example of code written in the past - this is all one function.

Dean Gladish
CS257-00
Prof. Amy Csizmar-Dalal

```
functionForDifferentModels <- function(modelname1, modelname2, groupname, foldername1, foldername2, realtimecomponentmodelspath,
iskdetest) {

# We take each of the files for the kcde model
kcdeFiles <- lapply(Sys.glob(paste(realtimecomponentmodelspath, foldername1, "/*.csv",
                    sep = "")),
            read.csv)


kcdeFiles <- lapply(Sys.glob(paste(realtimecomponentmodelspath, foldername2, "/*.csv", sep = "")),
            read.csv)


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

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


kcdeWeek1 <- kdeFiles
kcdeWeek2 <- kdeFiles
kcdeWeek3 <- kdeFiles
kcdeWeek4 <- kdeFiles
kcdeWeek1pointPredictionforILI <- kdeFiles
kcdeWeek2pointPredictionforILI <- kdeFiles
kcdeWeek3pointPredictionforILI <- kdeFiles
kcdeWeek4pointPredictionforILI <- 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")
```

The overarching problem is that all the helper functions are contained within this main function; the name 'functionForDifferentModels' is about the least descriptive that you could imagine, conveying neither its intended purpose nor its actual behavior, so everything has its unintended side effects and unanticipated behavior.

- The primary (functional) problem I ran into was the fact that you have to put the KDE model (this means kernel density estimation) into the second argument for modelname because otherwise you'll run into errors. Altogether the order of the model names mattering just makes this function extremely specific to this dataset.

- I should also talk about the structure of this function in order to (better) explain how it works; so, the first three arguments are there simply to denote the official names of the models and the name of the (research) group which was involved in creating them.

- What this function does, a bit of things. It was originally designed for finding the difference, using a k-s test, between influenza-prediction models developed by the Reich Lab and others for the CDC's FluSight initiative/contest.

```
  kdeWeek2pointPredictionforILI[[i]] <- USNationalIXWeeksAhead(kdeFiles[[i]],
                                      "week two", "yes")
  kdeWeek3pointPredictionforILI[[i]] <- USNationalIXWeeksAhead(kdeFiles[[i]],
                                      "week three", "yes")
  kdeWeek4pointPredictionforILI[[i]] <- USNationalIXWeeksAhead(kdeFiles[[i]],
                                      "week four", "yes")

}


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

removeStuff <- function(list, iskdetest) {
  # Removes elements 19, 20, 21, and 22.
  if (iskdetest=="iskdetest") {
    return(list.remove(list, c(19, 20, 21, 22)))
  }
  if (iskdetest=="isnotkdetest") {
    return(list)
  }

  # Since R works this way we can just do it all at once
}

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

kdeWeek1pointPredictionforILIReduced <- removeStuff(kdeWeek1pointPredictionforILI, iskdetest)
kdeWeek2pointPredictionforILIReduced <- removeStuff(kdeWeek2pointPredictionforILI, iskdetest)
kdeWeek3pointPredictionforILIReduced <- removeStuff(kdeWeek3pointPredictionforILI, iskdetest)
kdeWeek4pointPredictionforILIReduced <- removeStuff(kdeWeek4pointPredictionforILI, iskdetest)
# 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 <- kdeWeek1
temp2 <- kdeWeek2
temp3 <- kdeWeek3
temp4 <- kdeWeek4
temp5 <- kdeWeek1Reduced
temp6 <- kdeWeek2Reduced
temp7 <- kdeWeek3Reduced
temp8 <- kdeWeek4Reduced

# We also need to do this for our ILI values.
temp9 <- kdeWeek1pointPredictionforILI
temp10 <- kdeWeek2pointPredictionforILI
temp11 <- kdeWeek3pointPredictionforILI
temp12 <- kdeWeek4pointPredictionforILI
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.
  kdeWeek1[[i]] <- temp1[[((i+17)%%28)+1]]
  kdeWeek2[[i]] <- temp2[[((i+17)%%28)+1]]
  kdeWeek3[[i]] <- temp3[[((i+17)%%28)+1]]
  kdeWeek4[[i]] <- temp4[[((i+17)%%28)+1]]

  kdeWeek1Reduced[[i]] <- temp5[[((i+17)%%28)+1]]
```

← forexample this uses a boolean for whether or not the argument passed in is a certain type of model, leading to the aforementioned error message if the KDE model isn't passed as the second model name (modelname2).

* The commenting could just as easily have been block form instead of using individual hashtags for each case; also, this function does a bunch of unnecessary stuff which is made conditional as shown above, furthermore it includes a lot of copy-pasted code and repeated, vaguely-named operations which could just as easily have been done using more concise, dplyr/plyr-based methods.

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

  kdeWeek1pointPredictionforILI[[i]] <- temp9[[((i+17)%%28)+1]]
  kdeWeek2pointPredictionforILI[[i]] <- temp10[[((i+17)%%28)+1]]
  kdeWeek3pointPredictionforILI[[i]] <- temp11[[((i+17)%%28)+1]]
  kdeWeek4pointPredictionforILI[[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)
 }

 kdeWeek1CDF <- kdeWeek1
 kdeWeek2CDF <- kdeWeek2
 kdeWeek3CDF <- kdeWeek3
 kdeWeek4CDF <- kdeWeek4

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

 for (i in 1:28) {
  for (j in 1:131) {
   kdeWeek1CDF[[i]]$Value[j] <- sum(kdeWeek1[[i]]$Value[0:j])
   kdeWeek2CDF[[i]]$Value[j] <- sum(kdeWeek2[[i]]$Value[0:j])
   kdeWeek3CDF[[i]]$Value[j] <- sum(kdeWeek3[[i]]$Value[0:j])
   kdeWeek4CDF[[i]]$Value[j] <- sum(kdeWeek4[[i]]$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])
  }
 }
```

- We're basically rotating the order in which our files are stored within an array; not only does this diminish the reader's confidence in the accuracy of the code (although this does implement the Kolmogorov-Smirnov test accurately), but it causes the entire function to break down and become uneditable if we ever want to adapt it to compare other predictive models, also.

```
  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_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))


  # 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
```

- Also the other thing I would say is that the source for the data is not well-documented & following the links mentioned going to the ILI Net website and trying to download the dataset leads to an updated 2019-20 edition which is rendered incompatible due to some small error in this function, and yes this is all one function

causes us to be more confused about how to find the data.

Again, the local scope has these great plots but they are not well-named.

```
# 0.217 and 0.255 are matched by the first graph (for one week
# ahead) - in the real-time-component models folder,
# kdeFiles[[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(kdeFiles[[19]], "week one", "no")
tmp2 <- USNationalXWeeksAhead(kdeFiles[[23]], "week one", "no")
tmp3 <- USNationalXWeeksAhead(kdeFiles[[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 %ILI",
       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 %ILI",
       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, 1.79983, 1.88999, 2.06099, 2.09090,
2.28197, 2.28786, 2.42157, 2.49534, 2.58516, 2.77608, 3.19245, 3.37109, 3.69947, 4.73950, 4.96989, 5.71576, 5.75997, 5.90718, 6.45058,
6.52457, 7.17131, 7.39126, 7.52959), labels = c("1.5", "1.5", "1.6", "1.6", "1.7", "1.8", "1.9", "2.1", "2.1", "2.3", "2.3", "2.4", "2.5", "2.6", "2.8",
"3.2", "3.4", "3.7", "4.7", "5.0", "5.7", "5.8", "5.9", "6.5", "6.5", "7.2", "7.4", "7.5")) +
 theme(text = element_text(size = 9))


# It looks like the maximum distance is much
# more for the first three starting weeks
```

- Basically the results aren't as reproducible because the graphics (logo.svg etc.) depend on files which obviously the casual reader cannot even access.

```
# 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_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)) + geom_hline(aes(yintercept = 0.1422058)) + geom_text(aes(0,0.1422058,label = 0.01, vjust = -1)) +
geom_hline(aes(yintercept = 0.1186577)) + geom_text(aes(0,0.1186577,label = 0.05, vjust = -0.5)) + geom_hline(aes(yintercept = 0.1069283))+
geom_text(aes(0,0.1069283,label = 0.1, vjust = 1))



# 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 %ILI",
        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 %ILI",
        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, 1.79983, 1.88999, 2.06099, 2.09090,
2.28197, 2.28786, 2.42157, 2.49534, 2.58516, 2.77608, 3.19245, 3.37109, 3.69947, 4.73950, 4.96989, 5.71576, 5.75997, 5.90718, 6.45058,
6.52457, 7.17131, 7.39126, 7.52959), labels = c("1.5", "1.5", "1.6", "1.6", "1.7", "1.8", "1.9", "2.1", "2.1", "2.3", "2.3", "2.4", "2.5", "2.6", "2.8",
"3.2", "3.4", "3.7", "4.7", "5.0", "5.7", "5.8", "5.9", "6.5", "6.5", "7.2", "7.4", "7.5")) +
```

```
 theme(text = element_text(size = 9))+ geom_hline(aes(yintercept = 0.1422058)) + geom_text(aes(0,0.1422058,label = 0.01, vjust = -1)) +
geom_hline(aes(yintercept = 0.1186577)) + geom_text(aes(0,0.1186577,label = 0.05, vjust = -0.5)) + geom_hline(aes(yintercept = 0.1069283))+
geom_text(aes(0,0.1069283,label = 0.1, vjust = 1))


#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 IL1 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.

#Before we look for possible linearity we should create graphs of each individual distribution in order to determine how the models are
responsible for #the larger values of the first three points on the graphs.

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

# Because kdeweek1 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 kde1week
# 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) {
    kde1week[[j]][[i]] <- kdeWeek1[[j]]$Value[[i]]
    kde1week[[j]][[i]] <- kdeWeek1[[j]]$Value[[i]]
    kde2weeks[[j]][[i]] <- kdeWeek2[[j]]$Value[[i]]
    kde2weeks[[j]][[i]] <- kdeWeek2[[j]]$Value[[i]]
    kde3weeks[[j]][[i]] <- kdeWeek3[[j]]$Value[[i]]
    kde3weeks[[j]][[i]] <- kdeWeek3[[j]]$Value[[i]]
    kde4weeks[[j]][[i]] <- kdeWeek4[[j]]$Value[[i]]
    kde4weeks[[j]][[i]] <- kdeWeek4[[j]]$Value[[i]]
  }
}

# Now we must also define the plots.
max1 <- max(kde1week[[1]], kde1week[[2]], kde1week[[3]], kde1week[[4]], kde1week[[5]], kde1week[[6]])

# For some reason we have to define the legend not
# directly using the paste() command.
Legend <- paste("1w ", modelname1, sep = "")
plotw43AllWeeksAhead <- gf_line(kde1week[[1]] ~ seq_along(kde1week[[1]]), color = ~Legend, xlab = "Week 43", ylab = "") %>%
gf_line(kde1week[[1]] ~ seq_along(kde1week[[1]]), color = ~paste("1w ", modelname2, sep = "")) %>% gf_lims(y = c(0, max1)) %>%
```

Of course, it's well-documented with respect to the reason why I am doing these things.

- This whole function is definitely overcomplicated because if you're trying to compare different models, you have to make sure that the boolean "isKdetest" is changed. Changed, not to FALSE but to the string "isnotKdetest" because it's a boolean in string form — this controls a sub-function which runs only when the boolean is true.

```
gf_line(kde2weeks[[1]] ~ seq_along(kde2weeks[[1]]), color = ~paste("2w ", modelname1, sep = ""), xlab = "Week 43", ylab = "") %>%
gf_line(kde2weeks[[1]] ~ seq_along(kde2weeks[[1]]), color = ~paste("2w ", modelname2, sep = "")) %>% gf_lims(y = c(0, max1)) %>%
gf_line(kde3weeks[[1]] ~ seq_along(kde3weeks[[1]]), color = ~paste("3w ", modelname1, sep = ""), xlab = "Week 43", ylab = "") %>%
gf_line(kde3weeks[[1]] ~ seq_along(kde3weeks[[1]]), color = ~paste("3w ", modelname2, sep = "")) %>% gf_lims(y = c(0, max1)) %>%
gf_line(kde4weeks[[1]] ~ seq_along(kde4weeks[[1]]), color = ~paste("4w ", modelname1, sep = ""), xlab = "Week 43", ylab = "Probability for
Bins", title = "PMFs") %>%
gf_line(kde4weeks[[1]] ~ seq_along(kde4weeks[[1]]), color = ~paste("4w ", modelname2, sep = "")) %>% gf_lims(y = c(0, max1)) +
scale_x_continuous(breaks = c(0, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, 120, 130), labels = c("0%", "1%", "2%", "3%", "4%", "5%", "6%",
"7%", "8%", "9%", "10%", "11%", "12%", "13%")) + theme(text = element_text(size = 9))
Legend <- paste("1w ", modelname1, sep = "")
plotw44AllWeeksAhead <- gf_line(kde1week[[2]] ~ seq_along(kde1week[[2]]), color = ~Legend, xlab = "Week 44", ylab = "") %>%
gf_line(kde1week[[2]] ~ seq_along(kde1week[[2]]), color = ~paste("1w ", modelname2, sep = "")) %>% gf_lims(y = c(0, max1)) %>%
gf_line(kde2weeks[[2]] ~ seq_along(kde2weeks[[2]]), color = ~paste("2w ", modelname1, sep = ""), xlab = "Week 44", ylab = "") %>%
gf_line(kde2weeks[[2]] ~ seq_along(kde2weeks[[2]]), color = ~paste("2w ", modelname2, sep = "")) %>% gf_lims(y = c(0, max1)) %>%
gf_line(kde3weeks[[2]] ~ seq_along(kde3weeks[[2]]), color = ~paste("3w ", modelname1, sep = ""), xlab = "Week 44", ylab = "") %>%
gf_line(kde3weeks[[2]] ~ seq_along(kde3weeks[[2]]), color = ~paste("3w ", modelname2, sep = "")) %>% gf_lims(y = c(0, max1)) %>%
gf_line(kde4weeks[[2]] ~ seq_along(kde4weeks[[2]]), color = ~paste("4w ", modelname1, sep = ""), xlab = "Week 44", ylab = "Probability for
Bins", title = "PMFs") %>%
gf_line(kde4weeks[[2]] ~ seq_along(kde4weeks[[2]]), color = ~paste("4w ", modelname2, sep = "")) %>% gf_lims(y = c(0, max1)) +
scale_x_continuous(breaks = c(0, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, 120, 130), labels = c("0%", "1%", "2%", "3%", "4%", "5%", "6%",
"7%", "8%", "9%", "10%", "11%", "12%", "13%")) + theme(text = element_text(size = 9))
Legend <- paste("1w ", modelname1, sep = "")
plotw45AllWeeksAhead <- gf_line(kde1week[[3]] ~ seq_along(kde1week[[3]]), color = ~Legend, xlab = "Week 45", ylab = "") %>%
gf_line(kde1week[[3]] ~ seq_along(kde1week[[3]]), color = ~paste("1w ", modelname2, sep = "")) %>% gf_lims(y = c(0, max1)) %>%
gf_line(kde2weeks[[3]] ~ seq_along(kde2weeks[[3]]), color = ~paste("2w ", modelname1, sep = ""), xlab = "Week 45", ylab = "") %>%
gf_line(kde2weeks[[3]] ~ seq_along(kde2weeks[[3]]), color = ~paste("2w ", modelname2, sep = "")) %>% gf_lims(y = c(0, max1)) %>%
gf_line(kde3weeks[[3]] ~ seq_along(kde3weeks[[3]]), color = ~paste("3w ", modelname1, sep = ""), xlab = "Week 45", ylab = "") %>%
gf_line(kde3weeks[[3]] ~ seq_along(kde3weeks[[3]]), color = ~paste("3w ", modelname2, sep = "")) %>% gf_lims(y = c(0, max1)) %>%
gf_line(kde4weeks[[3]] ~ seq_along(kde4weeks[[3]]), color = ~paste("4w ", modelname1, sep = ""), xlab = "Week 45", ylab = "Probability for
Bins", title = "PMFs") %>%
gf_line(kde4weeks[[3]] ~ seq_along(kde4weeks[[3]]), color = ~paste("4w ", modelname2, sep = "")) %>% gf_lims(y = c(0, max1)) +
scale_x_continuous(breaks = c(0, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, 120, 130), labels = c("0%", "1%", "2%", "3%", "4%", "5%", "6%",
"7%", "8%", "9%", "10%", "11%", "12%", "13%")) + theme(text = element_text(size = 9))
Legend <- paste("1w ", modelname1, sep = "")
plotw46AllWeeksAhead <- gf_line(kde1week[[4]] ~ seq_along(kde1week[[4]]), color = ~Legend, xlab = "Week 46", ylab = "") %>%
gf_line(kde1week[[4]] ~ seq_along(kde1week[[4]]), color = ~paste("1w ", modelname2, sep = "")) %>% gf_lims(y = c(0, max1)) %>%
gf_line(kde2weeks[[4]] ~ seq_along(kde2weeks[[4]]), color = ~paste("2w ", modelname1, sep = ""), xlab = "Week 46", ylab = "") %>%
gf_line(kde2weeks[[4]] ~ seq_along(kde2weeks[[4]]), color = ~paste("2w ", modelname2, sep = "")) %>% gf_lims(y = c(0, max1)) %>%
gf_line(kde3weeks[[4]] ~ seq_along(kde3weeks[[4]]), color = ~paste("3w ", modelname1, sep = ""), xlab = "Week 46", ylab = "") %>%
gf_line(kde3weeks[[4]] ~ seq_along(kde3weeks[[4]]), color = ~paste("3w ", modelname2, sep = "")) %>% gf_lims(y = c(0, max1)) %>%
gf_line(kde4weeks[[4]] ~ seq_along(kde4weeks[[4]]), color = ~paste("4w ", modelname1, sep = ""), xlab = "Week 46", ylab = "Probability for
Bins", title = "PMFs") %>%
gf_line(kde4weeks[[4]] ~ seq_along(kde4weeks[[4]]), color = ~paste("4w ", modelname2, sep = "")) %>% gf_lims(y = c(0, max1)) +
scale_x_continuous(breaks = c(0, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, 120, 130), labels = c("0%", "1%", "2%", "3%", "4%", "5%", "6%",
"7%", "8%", "9%", "10%", "11%", "12%", "13%")) + theme(text = element_text(size = 9))
Legend <- paste("1w ", modelname1, sep = "")
plotw47AllWeeksAhead <- gf_line(kde1week[[5]] ~ seq_along(kde1week[[5]]), color = ~Legend, xlab = "Week 47", ylab = "") %>%
gf_line(kde1week[[5]] ~ seq_along(kde1week[[5]]), color = ~paste("1w ", modelname2, sep = "")) %>% gf_lims(y = c(0, max1)) %>%
gf_line(kde2weeks[[5]] ~ seq_along(kde2weeks[[5]]), color = ~paste("2w ", modelname1, sep = ""), xlab = "Week 47", ylab = "") %>%
gf_line(kde2weeks[[5]] ~ seq_along(kde2weeks[[5]]), color = ~paste("2w ", modelname2, sep = "")) %>% gf_lims(y = c(0, max1)) %>%
gf_line(kde3weeks[[5]] ~ seq_along(kde3weeks[[5]]), color = ~paste("3w ", modelname1, sep = ""), xlab = "Week 47", ylab = "") %>%
gf_line(kde3weeks[[5]] ~ seq_along(kde3weeks[[5]]), color = ~paste("3w ", modelname2, sep = "")) %>% gf_lims(y = c(0, max1)) %>%
gf_line(kde4weeks[[5]] ~ seq_along(kde4weeks[[5]]), color = ~paste("4w ", modelname1, sep = ""), xlab = "Week 47", ylab = "Probability for
Bins", title = "PMFs") %>%
gf_line(kde4weeks[[5]] ~ seq_along(kde4weeks[[5]]), color = ~paste("4w ", modelname2, sep = "")) %>% gf_lims(y = c(0, max1)) +
scale_x_continuous(breaks = c(0, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, 120, 130), labels = c("0%", "1%", "2%", "3%", "4%", "5%", "6%",
"7%", "8%", "9%", "10%", "11%", "12%", "13%")) + theme(text = element_text(size = 9))
Legend <- paste("1w ", modelname1, sep = "")
plotw48AllWeeksAhead <- gf_line(kde1week[[6]] ~ seq_along(kde1week[[6]]), color = ~Legend, xlab = "Week 48", ylab = "") %>%
gf_line(kde1week[[6]] ~ seq_along(kde1week[[6]]), color = ~paste("1w ", modelname2, sep = "")) %>% gf_lims(y = c(0, max1)) %>%
gf_line(kde2weeks[[6]] ~ seq_along(kde2weeks[[6]]), color = ~paste("2w ", modelname1, sep = ""), xlab = "Week 48", ylab = "") %>%
gf_line(kde2weeks[[6]] ~ seq_along(kde2weeks[[6]]), color = ~paste("2w ", modelname2, sep = "")) %>% gf_lims(y = c(0, max1)) %>%
gf_line(kde3weeks[[6]] ~ seq_along(kde3weeks[[6]]), color = ~paste("3w ", modelname1, sep = ""), xlab = "Week 48", ylab = "") %>%
gf_line(kde3weeks[[6]] ~ seq_along(kde3weeks[[6]]), color = ~paste("3w ", modelname2, sep = "")) %>% gf_lims(y = c(0, max1)) %>%
gf_line(kde4weeks[[6]] ~ seq_along(kde4weeks[[6]]), color = ~paste("4w ", modelname1, sep = ""), xlab = "Week 48", ylab = "Probability for
Bins", title = "PMFs") %>%
gf_line(kde4weeks[[6]] ~ seq_along(kde4weeks[[6]]), color = ~paste("4w ", modelname2, sep = "")) %>% gf_lims(y = c(0, max1)) +
scale_x_continuous(breaks = c(0, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, 120, 130), labels = c("0%", "1%", "2%", "3%", "4%", "5%", "6%",
"7%", "8%", "9%", "10%", "11%", "12%", "13%")) + theme(text = element_text(size = 9))
```

• This function literally does so many different things (I made it as a function so that we can compare different types of predictive models for influenza incidence such as creating beautiful graphs, deleting models' files based on a dysfunctionally-documented boolean and calculating the difference in the cumulative distribution functions generated from each models' predictions, that it's essentially not at all adaptable to be used by other researcher - this is the main issue and it came from not following any of the conventions/recommendations set forth by Robert Martin in Clean Code.

```
# 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, 2.09090, 2.28197, 2.28786,
2.42157, 2.49534, 2.58516, 2.77608, 3.19245, 3.37109, 3.69947, 4.73950, 4.96989, 5.71576, 5.75997, 5.90718, 6.45058, 6.52457, 7.17131,
7.39126, 7.52959), 1)

#######################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, df = 1))
#######################}
#######################anova(lineOfBestFit, stepAIC(lineOfBestFit), test = "Chisq")
#######################lineOfBestFit$residuals
#######################residualPlots(lineOfBestFit, tests = TRUE)
#######################influenceIndexPlot(lineOfBestFit)
#######################summary(lineOfBestFit)
#######################qqnorm(lineOfBestFit$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(kdeWeek1pointPredictionforILI[[i]]$Value -
                        kdeWeek1pointPredictionforILIReduced[[i]]$Value))
 pointValues[[2]][i] <- max(abs(kdeWeek2pointPredictionforILI[[i]]$Value -
                        kdeWeek2pointPredictionforILIReduced[[i]]$Value))
 pointValues[[3]][i] <- max(abs(kdeWeek3pointPredictionforILI[[i]]$Value -
                        kdeWeek3pointPredictionforILIReduced[[i]]$Value))
 pointValues[[4]][i] <- max(abs(kdeWeek4pointPredictionforILI[[i]]$Value -
                        kdeWeek4pointPredictionforILIReduced[[i]]$Value))
}

max2 <- max(pointValues[[1]], pointValues[[2]], pointValues[[3]], pointValues[[4]])

allPointValueDifferences <- gf_line(pointValues[[1]] ~ seq_along(pointValues[[1]]), color = ~"1w ahead",
                        xlab = "Index", ylab = "Difference in Predicted ILI") %>% gf_lims(y = c(0, max2)) %>%
 gf_line(pointValues[[2]] ~ seq_along(pointValues[[2]]), color = ~"2w ahead",
                        xlab = "Index", ylab = "Difference in Predicted ILI") %>% gf_lims(y = c(0, max2)) %>%
gf_line(pointValues[[3]] ~ seq_along(pointValues[[3]]), color = ~"3w ahead",
```

```
                    xlab = "Index", ylab = "Difference in Predicted ILI") %>% gf_lims(y = c(0, max2)) %>%
  gf_line(pointValues[[4]] ~ seq_along(pointValues[[4]]), color = ~"4w ahead",
                    xlab = "Index", ylab = "Difference in Predicted ILI", title = "Difference in Point Values") + theme(text =
element_text(size = 9))


grid.arrange(trueILIPlot, allPointValueDifferences)


grid.arrange(plotw43AllWeeksAhead, plotw44AllWeeksAhead, plotw45AllWeeksAhead, plotw46AllWeeksAhead, plotw47AllWeeksAhead,
plotw48AllWeeksAhead, top = paste(modelname1, " and ", modelname2, " PMFS for %ILI, All Weeks Ahead"))

grid.arrange(ksStatsAllWeeksAhead,
        linearityTestMaximums, actualILIagainstMaximums, top = paste("Comparing ", groupname, "'s ", modelname1,
" and ", modelname2, " Models \n for Selected Weeks of 2017-2018 (Week 43, 2017 to ",
"Week 18, 2018) \n Maximum Differences between the ", modelname1, " and ", modelname2, " Models' \n",
"Probability Mass Functions for All ILI Bins (increment 0.1)", sep = ""))


grid.arrange(trueKSStatsAllWeeksAhead, linearityTestKSStatistics, actualILIagainstKSStatistics, top = paste("Comparing ", groupname, "'s ",
modelname1,
" and ", modelname2, " Models \n for Selected Weeks of 2017-2018 (Week 43, 2017 to ",
"Week 18, 2018) \n Kolmogorov-Smirnov Test Statistics between the ", modelname1, " and ",
modelname2, " Models' \n Empirical CDF for All ILI Bins (increment 0.1)", sep = ""))

}
```