```
############ METHODS ###############

# Packages
library(dplyr)       # Data manipulation (0.8.0.1)
library(fBasics)     # Summary statistics (3042.89)
library(corrplot)    # Correlations (0.84)
library(psych)       # Correlation p-values (1.8.12)
library(grf)         # Generalized random forests (0.10.2)
library(rpart)       # Classification and regression trees, or CART (4.1-13)
library(rpart.plot)  # Plotting trees (3.0.6)
library(treeClust)   # Predicting leaf position for causal trees (1.1-7)
library(car)         # linear hypothesis testing for causal tree (3.0-2)
library(devtools)    # Install packages from github (2.0.1)
library(readr)       # Reading csv files (1.3.1)
library(tidyr)       # Database operations (0.8.3)
library(tibble)      # Modern alternative to data frames (2.1.1)
library(knitr)       # RMarkdown (1.21)
library(kableExtra)  # Prettier RMarkdown (1.0.1)
library(ggplot2)     # general plotting tool (3.1.0)
library(haven)       # read stata files (2.0.0)
library(aod)         # hypothesis testing (1.3.1)
library(evtree)      # evolutionary learning of globally optimal trees (1.0-7)
library(haven)
library(data.table)
library(caret)
library(magick)
library(xtable)
library(stargazer)
library(forecast)
library(trend)
library(lmtest)
library(nnfor)
library(tsutils)
library(fANCOVA)
library(kedd)
library(smooth)
library(Mcomp)
library(purrr)
library(rlang)
library(tsfknn)

# Run path
if (dir.exists("/Users/paula/stats205_project/")) {
  setwd("/Users/paula/stats205_project/")
  outputpath <- "/Users/paula/stats205_project/output/plots"
} else {
  setwd("/Users/tiffanyc/stats205_project/")
}

# load, clean, run descripties
source("code/load_clean.R")
source("code/descriptives.R")
```

```r
# fix colors
cols <- c("steelblue4", "goldenrod2", "grey4")

covs <- c("season", "yr", "mnth", "holiday", "weekday",
          "workingday", "weathersit", "temp", "atemp", "hum", "windspeed")

######## TEST & TRAIN DATA ############
set.seed(1)

#  test = last month
day <- day[order(day$dteday),]
day_train_forecast <- day[dteday <= "2012-11-30", ]
day_train_forecast <- day_train_forecast[order(day_train_forecast$dteday),]
day_test_forecast <- day[dteday > "2012-11-30", ]
day_test_forecast <- day_test_forecast[order(day_test_forecast$dteday),]

day_train_forecast[, index := .I]
day_test_forecast[, index := .I]

######### SIMPLE MOVING AVERAGE ############

# 15 fold CV is aproximately equal to a month in each bin
numbins <- 15
day_train_forecast$bins <- as.numeric(cut(day_train_forecast$dteday, numbins + 1))

# I ran it until 700
# only 20 for speed now since best was 15
order_max <- 20
mse_order <- matrix(nrow = order_max, ncol = 2)

for(o in 1:order_max) {

  mse_cv <- c()

  for(i in 1:numbins) {

    train <- day_train_forecast[bins <= i, ]
    test <- day_train_forecast[bins == i + 1, ]

    # run simple moving average on test set
    # hold out set is length of training set
    mod <- sma(train$cnt, order = o, h = nrow(test), interval = "nonparametric")

    # calcualte mse
    mse_cv[i] <- mean((test$cnt -  forecast(mod, h = nrow(test))$mean[1:nrow(test)])^2)


  }

  mse_order[o, 1] <- o
  mse_order[o, 2] <- mean(mse_cv)
}
```

```r
# which is the best simple moving average?
best_cv_order <- which.min(mse_order[, 2])

# refit with best cv order (and predict on test set)
best_sma <- sma(day_train_forecast$cnt, o = best_cv_order,
                h = nrow(day_test_forecast), interval = "nonparametric")
fcast_sma <- forecast(best_sma, h = nrow(day_test_forecast), interval = "nonparametric")

# plot results
results <- data.table(x = day$dteday)
results[, y := day$cnt]
results[, test := ifelse(x <= "2012-11-30", 0, 1)]
results[test == 0, fitted := best_sma$fitted]
results[test == 1, forecast := fcast_sma$mean[1:nrow(day_test_forecast)]]
results[test == 1, lower_ci := fcast_sma$lower[1:nrow(day_test_forecast)]]
results[test == 1, upper_ci := fcast_sma$upper[1:nrow(day_test_forecast)]]

mse_sma_test <- mean((day_test_forecast$cnt -
                        fcast_sma$mean[1:nrow(day_test_forecast)])^2)
mse_sma_train <- mean((day_train_forecast$cnt - best_sma$fitted)^2)



results %>%
  ggplot(aes(x = x, y = y)) + geom_point(size = 0.2) +
  geom_line(aes(x = x, y = fitted, col = "Fitted")) +
  geom_line(aes(x = x, y = forecast, col = "Forecast")) +
  geom_ribbon(aes(x = x, ymax = upper_ci, ymin = lower_ci, fill = ""), alpha = 0.3) +
  labs(title = paste0("Moving Average of Order ", best_cv_order),
       subtitle = paste0("Test RMSE: ", round(sqrt(mse_sma_test)),
                         "; Training RMSE: ", round(sqrt(mse_sma_train))),
       x = "Date", y = "Bike Count") +
  theme_linedraw() +
  theme(axis.ticks = element_blank()) +
  scale_color_manual(name="", values=c(cols[[1]],cols[[2]]),
                     labels=c("Training","Forecast")) +
  scale_fill_manual("Non-parametric 95% PI", values = "grey12")
ggsave(paste0(outputpath, "/sma_forecast.png"), width = 8, height = 4)


################ LOESS ##################


################ ... stl forecast ##############
ts_cnt <- msts(day$cnt, seasonal.periods=c(7, 365.25))
loess_decomp <- mstl(ts_cnt, lambda = 0)
png(filename="output/plots/mstl_whole_data.png")
plot(loess_decomp, main = "Seasonal (Weekly, Yearly) Decomposition
     using LOESS on Entire Dataset")
dev.off()

# seasonal decomposition using loess
# takes seasonality into account
```

```r
ts_cnt <- msts(day_train_forecast$cnt, seasonal.periods=c(7))
loess_decomp <- mstl(ts_cnt, s.window = 7, lambda = 0)

png(filename="output/plots/mstl_training_data.png")
plot(loess_decomp, main = "Seasonal (Weekly) Decomposition using
     LOESS on Training Data")
dev.off()

# forecast using random walk assumption
loess_forecast <- forecast(loess_decomp, h = nrow(day_test_forecast),
                           method = "naive")

exp_value <- function(x) {
  return(exp(x))
}

results <- data.table(x = day$dteday)
results[, y := day$cnt]
results[, test := ifelse(x <= "2012-11-30", 0, 1)]
results[test == 0, fitted := lapply(loess_decomp[, 2], exp)]
results[test == 1,
        forecast := unlist(lapply(loess_forecast$mean[1:nrow(day_test_forecast)],
                                  function(x) sapply(x, exp_value)))]

results[test == 1,
        upper_ci :=  unlist(lapply(loess_forecast$upper[1:nrow(day_test_forecast)],
                                   function(x) sapply(x, exp_value)))]

results[test == 1,
        lower_ci :=  unlist(lapply(loess_forecast$lower[1:nrow(day_test_forecast)],
                                   function(x) sapply(x, exp_value)))]

mse_stl_test <- mean((day_test_forecast$cnt -
                        unlist(lapply(loess_forecast$mean[1:nrow(day_test_forecast)],
                                                          function(x)
                                                            sapply(x, exp_value))))^2,
                     na.rm = TRUE)
mse_stl_train <- mean((day_train_forecast$cnt -
                         results[test == 0,]$fitted)^2, na.rm = TRUE)


results %>%
  ggplot(aes(x = x, y = y)) + geom_point(size = 0.2) +
  geom_line(aes(x = x, y = fitted, col = "Fitted")) +
  geom_line(aes(x = x, y = forecast, col = "Forecast")) +
  #geom_ribbon(aes(x = x, ymax = upper_ci, ymin = lower_ci, fill = ""), alpha = 0.3) +
  labs(title = paste0("Seasonal (Weekly) Decomposition using LOESS:
                       Forecast using Naive Random Walk"),
       subtitle = paste0("Test RMSE: ", round(sqrt(mse_stl_test)),
                         "; Training RMSE: ", round(sqrt(mse_stl_train))),
       x = "Date", y = "Bike Count") +
  theme_linedraw() +
  theme(axis.ticks = element_blank()) +
```

```
  scale_color_manual(name="", values=c(cols[[1]],cols[[2]]),
                     labels=c("Fitted Trend Cycle","Forecas Random Walkt")) +
  scale_fill_manual("95% PI", values = "grey12")
ggsave(paste0(outputpath, "/stl_loess_forecast_test.png"), width = 8, height = 4)

################ ... loess smoothing  #############

numbins <- 15
day_train_forecast$bins <- as.numeric(cut(day_train_forecast$dteday, numbins + 1))

# nax degree of polynomial
degree_max <- 2

mse_order <- matrix(nrow = degree_max + 1, ncol = 3)

for(d in 0:degree_max) {

  mse_cv <- c()

  for(i in 1:numbins) {

    train <- day_train_forecast[bins <= i, ]
    test <- day_train_forecast[bins == i + 1, ]

    # run on train; span is gcv
    loess_cv <- loess.as(train$instant,
                         train$cnt,
                         criterion = "gcv",
                         degree = d)
    loess_cv_span <- loess_cv$pars$span

    # predict to test data
    loess_test <- predict(loess_cv, test)
    # calcualte mse
    mse_cv[i] <- mean((test$cnt -  loess_test)^2)


  }

  mse_order[d + 1, 1] <- d
  mse_order[d + 1, 2] <- mean(mse_cv)
  mse_order[d + 1, 3] <- loess_cv_span

}

# which is the best simple moving average?
best_cv_order <- which.min(mse_order[, 2])

# which is the degree
best_degree <- which.min(mse_order[, 2]) - 1
best_span <- mse_order[best_degree + 1, 3]

# run on train
```

```r
day_train_forecast_loess <- day_train_forecast[, c("instant", "cnt")]
day_test_forecast_loess <- day_test_forecast[, c("instant", "cnt")]

loess_cv <- loess(cnt ~ instant,
                  data = day_train_forecast_loess,
                  span = best_span, degree = best_degree,
                  control=loess.control(surface="direct"))

# predict to test data
loess_test <- predict(loess_cv, day_test_forecast_loess)

results <- data.table(x = day$dteday)
results[, y := day$cnt]
results[, test := ifelse(x <= "2012-11-30", 0, 1)]
results[test == 1, fitted_test := loess_test]
results[test == 0, fitted_train := loess_cv$fitted]
results[, instant := day$instant]

mse_loess_test <- mean((results[test == 1, ]$y - results[test == 1, ]$fitted_test)^2)
mse_loess_train <- mean((results[test == 0, ]$y - results[test == 0, ]$fitted_train)^2)

results %>%
  ggplot(aes(x = x, y = y, col = "Datapoints")) +
  geom_point(size = 0.0000005, shape = 19)   +
  geom_line(aes(x = x, y = fitted_train, col = "Fitted")) +
  geom_line(aes(x = x, y = fitted_test, col = "Forecast")) +
  labs(x = "Date", y = "Bike Count") +
  theme_linedraw() +
  theme(axis.ticks = element_blank()) +
  theme(plot.title = element_text(hjust = 0.5, size = 8),
        plot.subtitle =  element_text(hjust = 0.5, size = 8),
        axis.text=element_text(size=10),
        axis.title = element_text(size=10),
        legend.text = element_text(size = 10)) +
  theme(legend.position = "bottom", legend.direction = "horizontal") +
  scale_colour_manual("",values = c("black",cols[[1]],cols[[2]]),
                      guide = guide_legend(override.aes =
                                             list(linetype = c("blank",
                                                               "solid",
                                                               "solid"),
                        shape = c(19, NA, NA))))
ggsave(paste0(outputpath, "/loess_smoothing.png"), width = 8, height = 4)

########## GAUSSIAN KERNEL ##################

########## ... based on index #############

day_train_forecast[, index := .I]

numbins <- 15
day_train_forecast$bins <- as.numeric(cut(day_train_forecast$dteday, numbins + 1))

# I ran it until 700
```

```r
# only 50 for speed now since best was 23
order_max <- 50

mse_order <- matrix(nrow = order_max, ncol = 2)

for(o in 1:order_max) {

  mse_cv <- c()

  for(i in 1:numbins) {

    train <- day_train_forecast[bins <= i, ]
    test <- day_train_forecast[bins == i + 1, ]

    # run simple moving average on test set
    # hold out set is length of training set
    mod <- ksmooth(train$index, train$cnt, "normal", bandwidth = o)
    pred_test <- predict(mod, h = nrow(test))

    # calcualte mse
    mse_cv[i] <- mean((test$cnt -  pred_test$y$mean[1:nrow(test)])^2)


  }

  mse_order[o, 1] <- o
  mse_order[o, 2] <- mean(mse_cv)
}

# which is the best simple moving average?
best_cv_order <- which.min(mse_order[, 2])

# refit with best cv order (and predict on test set)
best_gaussian <- ksmooth(day_train_forecast$index,
                         day_train_forecast$cnt,
                         "normal",
                         bandwidth = best_cv_order)

fcast_gaussian <- predict(best_gaussian, h = nrow(day_test_forecast))

# plot results
results <- data.table(x = day$dteday)
results[, y := day$cnt]
results[, test := ifelse(x <= "2012-11-30", 0, 1)]
results[test == 0, fitted := best_gaussian$y]
results[test == 1, forecast := fcast_gaussian$y$mean[1:nrow(day_test_forecast)]]
results[test == 1, lower_ci := fcast_gaussian$y$lower[1:nrow(day_test_forecast), 2]]
results[test == 1, upper_ci := fcast_gaussian$y$upper[1:nrow(day_test_forecast), 2]]

mse_gaussian_test <- mean((day_test_forecast$cnt -
                             fcast_gaussian$y$mean[1:nrow(day_test_forecast)])^2)
mse_gaussian_train <- mean((day_train_forecast$cnt - best_gaussian$y)^2)
```

```r
results %>%
  ggplot(aes(x = x, y = y)) + geom_point(size = 0.2) +
  geom_line(aes(x = x, y = fitted, col = "Fitted")) +
  geom_line(aes(x = x, y = forecast, col = "Forecast")) +
  geom_ribbon(aes(x = x, ymax = upper_ci, ymin = lower_ci, fill = ""), alpha = 0.3) +
  labs(title = paste0("Gausssian Kernel (CV) with Bandwidth ", best_cv_order),
       subtitle = paste0("Test RMSE: ", round(sqrt(mse_gaussian_test)),
                         "; Training RMSE: ", round(sqrt(mse_gaussian_train))),
       x = "Date", y = "Bike Count") +
  theme_linedraw() +
  theme(axis.ticks = element_blank()) +
  scale_color_manual(name="", values=c(cols[[1]],cols[[2]]),
                     labels=c("Training","Forecast")) +
  scale_fill_manual("95% PI", values = "grey12")
ggsave(paste0(outputpath, "/gaussian_kernel_predict.png"), width = 8, height = 4)


########## BOXCAR KERNEL ##################

########## ... based on index #############

numbins <- 15
day_train_forecast$bins <- as.numeric(cut(day_train_forecast$dteday, numbins + 1))

# I ran it until 200, best was 23
# only 30 for speed now
order_max <- 30

mse_order <- matrix(nrow = order_max, ncol = 2)

for(o in 1:order_max) {

  mse_cv <- c()

  for(i in 1:numbins) {

    train <- day_train_forecast[bins <= i, ]
    test <- day_train_forecast[bins == i + 1, ]

    # run simple moving average on test set
    # hold out set is length of training set
    mod <- ksmooth(train$index, train$cnt, "box", bandwidth = o)
    pred_test <- predict(mod, h = nrow(test))

    # calcualte mse
    mse_cv[i] <- mean((test$cnt -  pred_test$y$mean[1:nrow(test)])^2)


  }

  mse_order[o, 1] <- o
  mse_order[o, 2] <- mean(mse_cv)
}
```

```r
# which is the best simple moving average?
best_cv_order <- which.min(mse_order[, 2])

# refit with best cv order (and predict on test set)
best_boxcar <- ksmooth(day_train_forecast$index,
                       day_train_forecast$cnt,
                       "box",
                       bandwidth = best_cv_order)

fcast_boxcar <- predict(best_boxcar, h = nrow(day_test_forecast))

# plot results
results <- data.table(x = day$dteday)
results[, y := day$cnt]
results[, test := ifelse(x <= "2012-11-30", 0, 1)]
results[test == 0, fitted := best_boxcar$y]
results[test == 1, forecast := fcast_boxcar$y$mean[1:nrow(day_test_forecast)]]
results[test == 1, lower_ci := fcast_boxcar$y$lower[1:nrow(day_test_forecast), 2]]
results[test == 1, upper_ci := fcast_boxcar$y$upper[1:nrow(day_test_forecast), 2]]

mse_boxcar_test <- mean((day_test_forecast$cnt -
                           fcast_boxcar$y$mean[1:nrow(day_test_forecast)])^2)
mse_boxcar_train <- mean((day_train_forecast$cnt - best_boxcar$y)^2)


results %>%
  ggplot(aes(x = x, y = y)) + geom_point(size = 0.2) +
  geom_line(aes(x = x, y = fitted, col = "Fitted")) +
  geom_line(aes(x = x, y = forecast, col = "Forecast")) +
  geom_ribbon(aes(x = x, ymax = upper_ci, ymin = lower_ci, fill = ""), alpha = 0.3) +
  labs(title = paste0("Boxcar Kernel (CV) with Bandwidth ", best_cv_order),
       subtitle = paste0("Test RMSE: ", round(sqrt(mse_boxcar_test)),
                         "; Training RMSE: ", round(sqrt(mse_boxcar_train))),
       x = "Date", y = "Bike Count") +
  theme_linedraw() +
  theme(axis.ticks = element_blank()) +
  scale_color_manual(name="", values=c(cols[[1]],cols[[2]]),
                     labels=c("Training","Forecast")) +
  scale_fill_manual("95% PI", values = "grey12")
ggsave(paste0(outputpath, "/boxcar_kernel_predict.png"), width = 8, height = 4)


########## CUBIC SPLINES #################

# this does cross validation
# same as arima (0, 2, 2)
# shown how cubic smoothing splines can be used to obtain local
# linear forecasts for a univariate time series
ts_cnt <- msts(day_train_forecast$cnt, seasonal.periods=c(7, 365.25))
cubic_smoothing_spline <- splinef(ts_cnt,  h = nrow(day_test_forecast))
fcast_cubic_spline <- forecast(cubic_smoothing_spline,  h = nrow(day_test_forecast))
plot(fcast_cubic_spline)
```

```
results <- data.table(x = day$dteday)
results[, y := day$cnt]
results[, test := ifelse(x <= "2012-11-30", 0, 1)]
results[test == 0, fitted := cubic_smoothing_spline$fitted]
results[test == 1, forecast := fcast_cubic_spline$mean[1:nrow(day_test_forecast)]]
results[test == 1, upper_ci := fcast_cubic_spline$upper[1:nrow(day_test_forecast), 2]]
results[test == 1, lower_ci := fcast_cubic_spline$lower[1:nrow(day_test_forecast), 2]]

mse_cspline_test <- mean((day_test_forecast$cnt -
                            fcast_cubic_spline$mean[1:nrow(day_test_forecast)])^2)
mse_cspline_train <- mean((day_train_forecast$cnt - cubic_smoothing_spline$fitted)^2)

results %>%
  ggplot(aes(x = x, y = y)) + geom_point(size = 0.2) +
  geom_line(aes(x = x, y = fitted, col = "Fitted")) +
  geom_line(aes(x = x, y = forecast, col = "Forecast")) +
  geom_ribbon(aes(x = x, ymax = upper_ci, ymin = lower_ci, fill = ""), alpha = 0.3) +
  labs(title = paste0("Cubic Smoothing Spline "),
       subtitle = paste0("Test RMSE: ", round(sqrt(mse_cspline_test)),
                         "; Training RMSE: ", round(sqrt(mse_cspline_train))),
       x = "Date", y = "Bike Count") +
  theme_linedraw() +
  theme(axis.ticks = element_blank()) +
  scale_color_manual(name="", values=c(cols[[1]],cols[[2]]),
                     labels=c("Training","Forecast")) +
  scale_fill_manual("95% PI", values = "grey12")
ggsave(paste0(outputpath, "/cspline_forecast.png"), width = 8, height = 4)

######## EXPONENTIAL SMOOTHING ##############


####### ... univariate ###############
cross_validate_tbats <- function(b, tr, dtr, ar, nbin = numbins) {

  mse_cv <- c()

  for(i in 1:nbin) {


    train <- day_train_forecast[bins <= i, ]
    test <- day_train_forecast[bins == i + 1, ]

    ts_cnt <- msts(train$cnt, seasonal.periods=c(7, 365.25))
    mod <- tbats(ts_cnt, use.box.cox = b, use.trend = tr,
                 use.damped.trend = dtr, use.arma.errors = ar)

    mse_cv[i] <- mean((test$cnt -
                         forecast(mod, h = nrow(test))$mean[1:nrow(test)])^2)


  }
  return(mean(mse_cv))
}
```

```r
eval_true_matrix_input <- function(input) {
  if(input == TRUE) {
    return(1)
  } else {
    return(0)
  }
}


use_boxcox <- c(TRUE, FALSE)
use_trend <- c(TRUE, FALSE)
use_damped_trend <- c(TRUE, FALSE)
use_arma_errors <- c(TRUE, FALSE)

mse_tbats <- matrix(nrow = 16, ncol = 5)
c <- 0
for(b in use_boxcox) {
  for(tr in use_trend) {
    for(dtr in use_damped_trend) {
      for(ar in use_arma_errors) {

        c <- c + 1

        mse_tbats[c, 1] <- eval_true_matrix_input(b)
        mse_tbats[c, 2] <- eval_true_matrix_input(tr)
        mse_tbats[c, 3] <- eval_true_matrix_input(dtr)
        mse_tbats[c, 4] <- eval_true_matrix_input(ar)


        mse_tbats[c, 5] <- cross_validate_tbats(b = b, tr = tr, dtr = dtr, ar = ar)

      }
    }

  }

}

# what's the best model?
mse_tbats[which.min(mse_tbats[, 5]), ]

# refit best model on entire training set and forecast on test set
ts_cnt <- msts(day_train_forecast$cnt, seasonal.periods=c(7, 365.25))
best_tbats <- tbats(ts_cnt, use.trend = FALSE, use.damped.trend = TRUE,
                    use.arma.errors = FALSE)
forecast_mod_tbats <- forecast(best_tbats, h = nrow(day_test_forecast))

results <- data.table(x = day$dteday)
results[, y := day$cnt]
results[, test := ifelse(x <= "2012-11-30", 0, 1)]
results[test == 0, fitted := best_tbats$fitted.values]
results[test == 1, forecast := forecast_mod_tbats$mean[1:nrow(day_test_forecast)]]
results[test == 1, upper_ci := forecast_mod_tbats$upper[1:nrow(day_test_forecast)]]
```

```
results[test == 1, lower_ci := forecast_mod_tbats$lower[1:nrow(day_test_forecast)]]

mse_tbats_test <- mean((day_test_forecast$cnt -
                          forecast_mod_tbats$mean[1:nrow(day_test_forecast)])^2)
mse_tbats_train <- mean((day_train_forecast$cnt - best_tbats$fitted)^2)

results %>%
  ggplot(aes(x = x, y = y)) + geom_point(size = 0.2) +
  geom_line(aes(x = x, y = fitted, col = "Fitted")) +
  geom_line(aes(x = x, y = forecast, col = "Forecast")) +
  geom_ribbon(aes(x = x, ymax = upper_ci, ymin = lower_ci, fill = ""), alpha = 0.3) +
  labs(title = paste0("TBATS with Box-Cox,
                      Damped Trend and Weekly and Yearly Seasonality"),
       subtitle = paste0("Test RMSE: ", round(sqrt(mse_tbats_test)),
                         "; Training RMSE: ", round(sqrt(mse_tbats_train))),
       x = "Date", y = "Bike Count") +
  theme_linedraw() +
  theme(axis.ticks = element_blank()) +
  scale_color_manual(name="", values=c(cols[[1]],cols[[2]]),
                     labels=c("Training","Forecast")) +
  scale_fill_manual("95% PI", values = "grey12")
ggsave(paste0(outputpath, "/tbats_forecast.png"), width = 8, height = 4)

################ ARIMA ################

######### ... univariate ############

ts_cnt <- msts(day_train_forecast$cnt, seasonal.periods=c(7, 365.25))

best_sarima <- auto.arima(ts_cnt, D = 1)
fcast_sarima <- forecast(best_sarima, h = nrow(day_test_forecast))
plot(fcast_sarima)

results <- data.table(x = day$dteday)
results[, y := day$cnt]
results[, test := ifelse(x <= "2012-11-30", 0, 1)]
results[test == 0, fitted := best_sarima$fitted]
results[test == 1, forecast := fcast_sarima$mean[1:nrow(day_test_forecast)]]
results[test == 1, upper_ci := fcast_sarima$upper[1:nrow(day_test_forecast), 2]]
results[test == 1, lower_ci := fcast_sarima$lower[1:nrow(day_test_forecast), 2]]

mse_sarima_test <- mean((day_test_forecast$cnt -
                           fcast_sarima$mean[1:nrow(day_test_forecast)])^2)
mse_sarima_train <- mean((day_train_forecast$cnt - best_sarima$fitted)^2)

results %>%
  ggplot(aes(x = x, y = y)) + geom_point(size = 0.2) +
  geom_line(aes(x = x, y = fitted, col = "Fitted")) +
  geom_line(aes(x = x, y = forecast, col = "Forecast")) +
  geom_ribbon(aes(x = x, ymax = upper_ci, ymin = lower_ci, fill = ""), alpha = 0.3) +
  labs(title = paste0("Best (Seasonal) ARIMA: ", fcast_sarima$method),
       subtitle = paste0("Test RMSE: ", round(sqrt(mse_sarima_test)),
                         "; Training RMSE: ", round(sqrt(mse_sarima_train))),
```

```
       x = "Date", y = "Bike Count") +
  theme_linedraw() +
  theme(axis.ticks = element_blank()) +
  scale_color_manual(name="", values=c(cols[[1]],cols[[2]]),
                     labels=c("Training","Forecast")) +
  scale_fill_manual("95% PI", values = "grey12")
ggsave(paste0(outputpath, "/sarima_forecast.png"), width = 8, height = 4)


############## ... multivariate #################
best_sarima_covs <- auto.arima(ts_cnt,
                               xreg = as.matrix(day_train_forecast[, ..covs]), D = 1)

fcast_sarima_covs <- forecast(best_sarima_covs,
                              h = nrow(day_test_forecast),
                              xreg = as.matrix(day_test_forecast[, ..covs]))

results <- data.table(x = day$dteday)
results[, y := day$cnt]
results[, test := ifelse(x <= "2012-11-30", 0, 1)]
results[test == 0, fitted := best_sarima_covs$fitted]
results[test == 1, forecast := fcast_sarima_covs$mean[1:nrow(day_test_forecast)]]
results[test == 1, upper_ci := fcast_sarima_covs$upper[1:nrow(day_test_forecast), 2]]
results[test == 1, lower_ci := fcast_sarima_covs$lower[1:nrow(day_test_forecast), 2]]

write.csv(results, file = paste0(outputpath, "/sarima_results.csv"))

mse_sarimacovs_test <- mean((day_test_forecast$cnt -
                               fcast_sarima_covs$mean[1:nrow(day_test_forecast)])^2)
mse_sarimacovs_train <- mean((day_train_forecast$cnt - best_sarima_covs$fitted)^2)

results %>%
  ggplot(aes(x = x, y = y)) + geom_point(size = 0.2) +
  geom_line(aes(x = x, y = fitted, col = "Fitted")) +
  geom_line(aes(x = x, y = forecast, col = "Forecast")) +
  geom_ribbon(aes(x = x, ymax = upper_ci, ymin = lower_ci, fill = ""), alpha = 0.3) +
  labs(title = paste0("Best (Seasonal) ARIMA: ", fcast_sarima_covs$method),
       subtitle = paste0("Test RMSE: ", round(sqrt(mse_sarimacovs_test)),
                         "; Training RMSE: ", round(sqrt(mse_sarimacovs_train))),
       x = "Date", y = "Bike Count") +
  theme_linedraw() +
  theme(axis.ticks = element_blank()) +
  scale_color_manual(name="", values=c(cols[[1]],cols[[2]]),
                     labels=c("Training","Forecast")) +
  scale_fill_manual("95% PI", values = "grey12")
ggsave(paste0(outputpath, "/sarimacovs_forecast.png"), width = 8, height = 4)

results %>%
  dplyr::filter(x >= "2012-12-01") %>%
  ggplot(aes(x = x, y = y, col = "Data")) + geom_line()+
  geom_line(aes(x = x, y = forecast, col = "Forecast")) +
  labs(x = "Date", y = "Bike Count") +
  theme_linedraw() +
```

```
  theme(axis.ticks = element_blank()) +
  theme(plot.title = element_text(hjust = 0.5, size = 8),
        plot.subtitle =  element_text(hjust = 0.5, size = 8),
        axis.text=element_text(size=10),
        axis.title = element_text(size=10),
        legend.text = element_text(size = 10)) +
  theme(legend.position = "bottom", legend.direction = "horizontal") +
  scale_colour_manual("",values = c(cols[[1]],cols[[2]]),
                      guide = guide_legend(override.aes = list(linetype = c("solid",
                                                                            "solid"),
                                                               shape = c(NA, NA))))
ggsave(paste0(outputpath, "/sarima_forecast_include.png"), width = 8, height = 4)

################ Neural Network Autoregression ################

############### ... with covariates ##################
set.seed(1)
ts_cnt <- msts(day_train_forecast$cnt, seasonal.periods=c(7, 365.25))
nn_covs <- nnetar(ts_cnt, xreg = as.matrix(day_train_forecast[, ..covs]), repeats = 100)
nn_forecast_covs <- forecast(nn_covs,h=nrow(day_test_forecast),
                             xreg = as.matrix(day_test_forecast[, ..covs]))
plot(nn_forecast_covs)

results <- data.table(x = day$dteday)
results[, y := day$cnt]
results[, instant := day$instant]
results[, test := ifelse(x <= "2012-11-30", 0, 1)]
results[test == 0, fitted := nn_covs$fitted]
results[test == 1, forecast := nn_forecast_covs$mean[1:nrow(day_test_forecast)]]


mse_nn_test_covs <- mean((day_test_forecast$cnt -
                            nn_forecast_covs$mean[1:nrow(day_test_forecast)])^2)
mse_nn_train_covs <- mean((day_train_forecast[index >= 366, ]$cnt -
                             nn_covs$fitted[366:length(nn_covs$fitted)])^2)

results %>%
  ggplot(aes(x = x, y = y)) + geom_point(size = 0.2) +
  geom_line(aes(x = x, y = fitted, col = "Fitted")) +
  geom_line(aes(x = x, y = forecast, col = "Forecast")) +
  labs(title = paste0("Neural Network Autoregression: Feed-Forward NN;
                      Single Hidden Layer; With Covariates"),
       subtitle = paste0("Test RMSE: ", round(sqrt(mse_nn_test_covs)),
                         " Training MSE: ",  round(sqrt(mse_nn_train_covs))),
       x = "Date", y = "Bike Count") +
  theme_linedraw() +
  theme(axis.ticks = element_blank()) +
  scale_color_manual(name="", values=c(cols[[1]],cols[[2]]),
                     labels=c("Training","Forecast")) +
  scale_fill_manual("95% PI", values = "grey12")
ggsave(paste0(outputpath, "/nn_forecast_covariates.png"),
       width = 8, height = 4)
```

```r
############## ... without covariates ##############

set.seed(1)
ts_cnt <- msts(day_train_forecast$cnt, seasonal.periods=c(7, 365.25))
nn <- nnetar(ts_cnt, repeats = 10000)
nn_forecast <- forecast(nn,h=nrow(day_test_forecast))
plot(nn_forecast)

results <- data.table(x = day$dteday)
results[, y := day$cnt]
results[, instant := day$instant]
results[, test := ifelse(x <= "2012-11-30", 0, 1)]
results[test == 0, fitted := nn$fitted]
results[test == 1, forecast := nn_forecast$mean[1:nrow(day_test_forecast)]]

mse_nn_test <- mean((day_test_forecast$cnt -
                       nn_forecast$mean[1:nrow(day_test_forecast)])^2)
mse_nn_train <- mean((day_train_forecast[index >= 366, ]$cnt -
                        nn$fitted[366:length(nn$fitted)])^2)

results %>%
  ggplot(aes(x = x, y = y)) + geom_point(size = 0.2) +
  geom_line(aes(x = x, y = fitted, col = "Fitted")) +
  geom_line(aes(x = x, y = forecast, col = "Forecast")) +
  labs(title = paste0("Neural Network Autoregression: Feed-Forward NN;
                      Single Hidden Layer; Without Covariates"),
       subtitle = paste0("Test RMSE: ", round(sqrt(mse_nn_test)),
                         " Training MSE: ",  round(sqrt(mse_nn_train))),
       x = "Date", y = "Bike Count") +
  theme_linedraw() +
  theme(axis.ticks = element_blank()) +
  scale_color_manual(name="", values=c(cols[[1]],cols[[2]]),
                     labels=c("Training","Forecast")) +
  scale_fill_manual("95% PI", values = "grey12")
ggsave(paste0(outputpath, "/nn_forecast_no_covariates.png"), width = 8, height = 4)
```