Adv. Time Series Econometrics Assignment 3
Computational appendix

A. Schmidt, G. Mingoli, T. Nguyen

3/10/2021

Description

This code automatically downloads data from the web about the monthly quarterly rate of nonfarm payroll (EMP) from 1954 to 2020. The aim is to compare forecasts from different models. The methods used to estimate the models are:

1. A multiple linear regression of 9 covariates
2. A weighted average of 9 simple linear regressions
3. A factor augmented model
4. The peLASSO model from Diebold and Shin (2019)
5. The local linear forests from Friedberg et al (2020)
6. A simple AR(1) model
7. The historical mean

Results are displayed in three parts: 1. Graphs combining the actual series with the 1 step rolling window forecast 2. MSE, RMSE and MAE are computed for each method with and without the months in 2020 3. The Diebold Mariano test is computed to compare the different methods against the benchmark model

Imports and setup

We are using the `glmnet` package (available here) for implementing the peLASSO; the `dynlm` (available here) to estimate the factor model (together with the `PANICr` - downloaded directly from the CRAN archive because it has been deprecated) and the `grf` package for the local linear forest (available here). The installation of dependencies should run automatically in any computer.

```
rm(list=ls())         # Clean memory
graphics.off()        # Close graphs

# Install package 'PANICr' from source
pkgTest <- function(x)
{
  if (!require(x,character.only = TRUE))
  {
    packageurl <- "https://cran.r-project.org/src/contrib/Archive/PANICr/PANICr_1.0.0.tar.gz"
    install.packages(packageurl, repos = NULL, type = "source")
  } else
  {
```

```
    library(PANICr)
  }
}

pkgTest("PANICr")


# Verify if a package is installed, if not, download and install before loading.
chooseCRANmirror(graphics = FALSE, ind = 10)
if (!require("pacman")) install.packages("pacman")
pacman::p_load(reshape2, ggplot2, latex2exp, gridExtra, readxl, xts, Metrics, dynlm,
               forecast, zoo, multDM, httr, rio, kableExtra, stargazer, multDM, tibble,
               zoo, grf, glmnet, MCMCpack, lubridate, scales, RColorBrewer, knitr)
```

Data is directly downloadable from the web and loaded into R, so no further action here is needed.

```
# Downloading and reading data
urlRemote <- "https://github.com/aishameriane"
pathGithub <- "/Stuff/blob/master/ATSE/USEMP.xlsx?raw=TRUE"

url <- paste0(urlRemote, pathGithub)
dfData01 <- rio::import(url)
# dfData01[,1] <- as.yearmon(dfData01[,1])
df <- xts(dfData01[, -1], order.by = dfData01$Month)
```

The next chunck has some auxiliary functions that were modified from the `forecast` package to make our descriptive plots and also to make the table with the descriptive information from the data (which was supressed from the final version of the report.)

```
# Other functions from the forecast package ACF
# plot function adapted from the `forecast` package
autoplot.acf01 <- function(object, ci = 0.95, title = "EMP quaterly growth",
    ...) {
    if (!requireNamespace("ggplot2", quietly = TRUE)) {
        stop("ggplot2 is needed for this function to work.
         Install it via install.packages(\"ggplot2\")",
            call. = FALSE)
    } else {
        if (!inherits(object, "acf")) {
            stop("autoplot.acf requires a acf object, use object=object")
        }

        acf <- `dimnames<-`(object$acf, list(NULL,
            object$snames, object$snames))
        lag <- `dimnames<-`(object$lag, list(NULL,
            object$snames, object$snames))

        data <- as.data.frame.table(acf)[-1]
        data$lag <- as.numeric(lag)

        if (object$type == "correlation") {
            data <- data[data$lag != 0, ]
        }
```

```
    # Initialise ggplot object
    p <- ggplot2::ggplot(ggplot2::aes_(x = ~lag,
        xend = ~lag, y = 0, yend = ~Freq), data = data)
    p <- p + ggplot2::geom_hline(yintercept = 0)

    # Add data
    p <- p + ggplot2::geom_segment(lineend = "butt",
        ...)

    # Add ci lines (assuming white noise input)
    ci <- qnorm((1 + ci)/2)/sqrt(object$n.used)
    p <- p + ggplot2::geom_hline(yintercept = c(-ci,
        ci), colour = "blue", linetype = "dashed")

    # Add facets if needed
    if (any(dim(object$acf)[2:3] != c(1, 1))) {
        p <- p + ggplot2::facet_grid(as.formula(paste0(colnames(data)[1:2],
            collapse = "~")))
    }

    # Prepare graph labels
    if (!is.null(object$ccf)) {
        ylab <- "CCF"
        ticktype <- "ccf"
        # main <- paste('Series:', object$snames)
        main <- title
        nlags <- round(dim(object$lag)[1]/2)
    } else if (object$type == "partial") {
        ylab <- "PACF"
        ticktype <- "acf"
        # main <- paste('Series:', object$series)
        main <- title
        nlags <- dim(object$lag)[1]
    } else if (object$type == "correlation") {
        ylab <- "ACF"
        ticktype <- "acf"
        # main <- paste('Series:', object$series)
        main <- title
        nlags <- dim(object$lag)[1]
    } else {
        ylab <- NULL
    }

    # Add seasonal x-axis Change ticks to be seasonal
    # and prepare default title
    if (!is.null(object$tsp)) {
        freq <- object$tsp[3]
    } else {
        freq <- 1
    }
    if (!is.null(object$periods)) {
        periods <- object$periods
        periods <- periods[periods != freq]
```

```
            minorbreaks <- periods * seq(-20:20)
        } else {
            minorbreaks <- NULL
        }
        p <- p + ggplot2::scale_x_continuous(breaks = seasonalaxis(freq,
            nlags, type = ticktype, plot = FALSE),
            minor_breaks = minorbreaks)
        p <- p + ggAddExtras(ylab = ylab, xlab = "Lag",
            main = main)
        p <- p + ggplot2::theme(axis.title.x = element_text(size = 8),
            axis.title.y = element_text(size = 8),
            plot.title = element_text(size = 10))
        return(p)
    }
}


# Make nice horizontal axis with ticks at seasonal
# lags Return tick points if breaks=TRUE
seasonalaxis <- function(frequency, nlags, type, plot = TRUE) {
    # List of unlabelled tick points
    out2 <- NULL
    # Check for non-seasonal data
    if (length(frequency) == 1) {
        # Compute number of seasonal periods
        np <- trunc(nlags/frequency)
        evenfreq <- (frequency%%2L) == 0L

        # Defaults for labelled tick points
        if (type == "acf") {
            out <- pretty(1:nlags)
        } else {
            out <- pretty(-nlags:nlags)
        }

        if (frequency == 1) {
            if (type == "acf" && nlags <= 16) {
                out <- 1:nlags
            } else if (type == "ccf" && nlags <= 8) {
                out <- (-nlags:nlags)
            } else {
                if (nlags <= 30 && type == "acf") {
                  out2 <- 1:nlags
                } else if (nlags <= 15 && type == "ccf") {
                  out2 <- (-nlags:nlags)
                }
                if (!is.null(out2)) {
                  out <- pretty(out2)
                }
            }
        } else if (frequency > 1 && ((type == "acf" &&
            np >= 2L) || (type == "ccf" && np >= 1L))) {
            if (type == "acf" && nlags <= 40) {
```

```
                out <- frequency * (1:np)
                out2 <- 1:nlags
                # Add half-years
                if (nlags <= 30 && evenfreq && np <=
                  3) {
                  out <- c(out, frequency * ((1:np) -
                    0.5))
                }
            } else if (type == "ccf" && nlags <= 20) {
                out <- frequency * (-np:np)
                out2 <- (-nlags:nlags)
                # Add half-years
                if (nlags <= 15 && evenfreq && np <=
                  3) {
                  out <- c(out, frequency * ((-np:np) +
                    0.5))
                }
            } else if (np < (12 - 4 * (type == "ccf"))) {
                out <- frequency * (-np:np)
            }
        }
    } else {
        # Determine which frequency to show
        np <- trunc(nlags/frequency)
        frequency <- frequency[which(np <= 16)]
        if (length(frequency) > 0L) {
            frequency <- min(frequency)
        } else {
            frequency <- 1
        }
        out <- seasonalaxis(frequency, nlags, type,
            plot = FALSE)
    }
    if (plot) {
        axis(1, at = out)
        if (!is.null(out2)) {
            axis(1, at = out2, tcl = -0.2, labels = FALSE)
        }
    } else {
        return(out)
    }
}
ggPacf01 <- function(x, lag.max = NULL, plot = TRUE,
    na.action = na.contiguous, demean = TRUE, type = "correlation",
    ...) {
    object <- Acf(x, lag.max = lag.max, type = type,
        na.action = na.action, demean = demean, plot = FALSE)
    object$series <- deparse(substitute(x))
    if (plot) {
        return(autoplot(object, ...))
    } else {
        return(object)
    }
```

```
}
ggAddExtras <- function(xlab = NA, ylab = NA, main = NA) {
    dots <- eval.parent(quote(list(...)))
    extras <- list()
    if ("xlab" %in% names(dots) || is.null(xlab) ||
        any(!is.na(xlab))) {
        if ("xlab" %in% names(dots)) {
            extras[[length(extras) + 1]] <- ggplot2::xlab(dots$xlab)
        } else {
            extras[[length(extras) + 1]] <- ggplot2::xlab(paste0(xlab[!is.na(xlab)],
                collapse = " "))
        }
    }
    if ("ylab" %in% names(dots) || is.null(ylab) ||
        any(!is.na(ylab))) {
        if ("ylab" %in% names(dots)) {
            extras[[length(extras) + 1]] <- ggplot2::ylab(dots$ylab)
        } else {
            extras[[length(extras) + 1]] <- ggplot2::ylab(paste0(ylab[!is.na(ylab)],
                collapse = " "))
        }
    }
    if ("main" %in% names(dots) || is.null(main) ||
        any(!is.na(main))) {
        if ("main" %in% names(dots)) {
            extras[[length(extras) + 1]] <- ggplot2::ggtitle(dots$main)
        } else {
            extras[[length(extras) + 1]] <- ggplot2::ggtitle(paste0(main[!is.na(main)],
                collapse = " "))
        }
    }
    if ("xlim" %in% names(dots)) {
        extras[[length(extras) + 1]] <- ggplot2::xlim(dots$xlim)
    }
    if ("ylim" %in% names(dots)) {
        extras[[length(extras) + 1]] <- ggplot2::ylim(dots$ylim)
    }
    return(extras)
}
ggtsbreaks <- function(x) {
    # Make x axis contain only whole numbers (e.g.,
    # years)
    return(unique(round(pretty(floor(x[1]):ceiling(x[2])))))
}

# Function to build a summary descriptives table
desc <- function(x) {
    n <- length(x)
    minimum <- min(x, na.rm = TRUE)
    first_q <- quantile(x, 0.25, na.rm = TRUE)
    media <- mean(x, na.rm = TRUE)
    mediana <- median(x, na.rm = TRUE)
    third_q <- quantile(x, 0.75, na.rm = TRUE)
```

```
    maximum <- max(x, na.rm = TRUE)
    std <- sd(x, na.rm = TRUE)
    return(list(n = n, minimum = minimum, first_quar = first_q,
        media = media, mediana = mediana, third_quar = third_q,
        maximum = maximum, std = std))
}
```

The `rolling.red()` function is an auxiliary wrapper to roll the forecasting and estimation window automatically by using the `rollapply()` function.

```
# Rolling regression function
width = 181
rolling.reg <- function(df, model, width) {
    pred <- rollapply(data = zoo(df), width = width,
        FUN = model, by.column = FALSE, align = "right")

    return(as.data.frame(pred))
}
```

Estimation

The first model to be estimated is a multiple linear regression with all covariates against EMP plus an intercept.

```
# Model 1: linear regression
lin.reg <- function(df) {

    df <- as.data.frame(df)

    # Set in and out-of-sample
    df_in <- df[-nrow(df), ]
    df_out <- df[nrow(df), ]

    # Step 1: Estimate the coefficients
    fit <- lm(EMP ~ ., data = df_in)

    # Step 2: Construct the forecast
    pred <- predict(fit, df_out)

    return(pred)
}

# Rolling regression
pred1 <- rolling.reg(df = df, model = lin.reg, width)
```

The second model is an average of 9 simple regressions (one for each covariate).

```
# Model 2: Averaged regressions
avlin.reg <- function(df) {
    df = as.data.frame(df)
```

```
    pred2 = matrix(NA, 9)
    for (i in 1:9) {
        # Set in and out-of-sample
        df_in <- df[-nrow(df), c(1, i + 1)]
        df_out <- df[nrow(df), c(1, i + 1)]

        # Step 1: Estimate the coefficients
        fit <- lm(EMP ~ ., data = df_in)

        # Step 2: Construct the forecast
        pred2[i] <- predict(fit, df_out)
    }
    pred = mean(pred2)

    return(pred)
}

# Rolling regression
pred2 <- rolling.reg(df = df, model = avlin.reg, width)
```

For the third model, we estimated the factor-augmented regression. In general terms, we first get principal components (chosen optimally using the algorithm from Bai & Ng, 2003). Secondly, we estimate the model using the most relevant factors.

```
# Model 3: Factor-augmented regression
fac.aug <- function(df) {
    df <- as.xts(df)

    # Step 1: Compute the first r PCs
    X <- df[, -1]
    Fhat <- getnfac(X, kmax = ncol(X), criteria = "BIC3")$Fhat

    # Set in and out-of-sample
    df_new <- cbind(df[, 1], Fhat)
    df_in <- df_new[-nrow(df_new), ]
    df_out <- df_new[nrow(df_new), ]

    # Step 2: Estimate the coefficients
    fit <- dynlm(EMP ~ L(EMP, -1) + L(Fhat, -1) + 0,
        data = df_in)

    # Step 3: Construct the forecast
    pred <- predict(fit, df_out)

    return(pred)
}

# Rolling regression
pred3 <- rolling.reg(df = df, model = fac.aug, width)
```

The fourth model is the partially-egalitarian Lasso method from Diebold and Shin. We do the estimation in two steps: in the first one we estimate a lasso using EMT and the covariates to obtain the relevant betas for estimation. We then estimate the model again, but we replace

```r
# Model 4: PELASSO regression

pe.Lasso <- function(df) {

    iT = nrow(df)

    ##### First step
    x <- as.matrix(df[1:(iT - 1), 2:10])
    y <- as.matrix(df[1:(iT - 1), 1])
    lambda0 <- cv.glmnet(x, y)$lambda.min
    fit <- glmnet(x, y, family = "gaussian", alpha = 1,
        lambda = lambda0)
    beta <- coef(fit)
    ind <- beta@i
    ##### Second step
    if (length(ind) < 3) {
        ind = c(0, 1, 2)
    }

    x <- as.matrix(df[1:(iT - 1), ind])
    y <- as.matrix(df[1:(iT - 1), 1])
    fbar <- rowMeans(x)
    ybar <- y - fbar
    lambda0 <- cv.glmnet(x, ybar)$lambda.min
    fit <- glmnet(x, ybar, family = "gaussian", alpha = 1,
        lambda = lambda0)

    pred0 <- predict(fit, newx = t(df[iT, ind]))
    pred1 <- pred0 + mean(df[iT, ind])

    return(pred1)
}

# Rolling regression
pred4 <- rolling.reg(df = df, model = pe.Lasso, width)
```

The local linear forest uses the package developed by Friedberg et. al (2020). The function `ll_regression_forest` produces the fit and gives the necessary information to produce the forecasts.

```r
# Model 5: Local linear forest


ll_forest <- function(df) {

    # Step 0: Organize the objects
    df <- as.data.frame(df)
    dfEMP <- as.matrix(df[-nrow(df), 1])
    dfX <- as.matrix(df[-nrow(df), 2:ncol(df)])
    dfXnew <- as.matrix(df[nrow(df), 2:ncol(df)])

    # Step 1: Estimate the coefficients
    fit <- ll_regression_forest(X = dfX, Y = dfEMP,
        enable.ll.split = TRUE)
```

```
    # Step 2: Construct the forecast
    pred <- predict(fit, dfXnew, linear.correction.variables = 1)$predictions

    return(as.numeric(pred))
}

pred5 <- rolling.reg(df = df, model = ll_forest, width)
```

The AR(1) model is simply using a lag of the variable `EMP` and an intercept as covariates. This is used as benchmark.

```
########################## Model 6: AR(1)
ar1 <- function(df) {
    df <- as.xts(df)

    # Set in and out-of-sample
    df_in <- df[-nrow(df), 1]
    df_out <- df[nrow(df), 1]

    # Step 1: Estimate the coefficients
    fit <- dynlm(EMP ~ L(EMP, -1), data = df_in)

    # Step 2: Construct the forecast
    pred <- predict(fit, df_out)

    return(pred)
}

# Rolling regression
pred6 <- rolling.reg(df = df, model = ar1, width)
```

The last model is our second benchmark and corresponds to the average of the values for `EMP` using the observations before the forecast period.

```
############################## Model 7: historical average
hist_ave <- function(df) {

    pred <- mean(head(df[, 1], -1))

    return(as.numeric(pred))
}

# Rolling regression
pred7 <- rolling.reg(df = df, model = hist_ave, width)
```

Descriptive analysis

We ran some basic descriptive analysis for sanity check of the data. With exception of the graphs, the tables are not part of our main report.

Table 1: Table 1 - First and last observations of the dataset

EMP	EMPL	HOURS	CLAIMS	ORDERSC	ORDERSI	ORDERSN	BUILDING	STOCK	SPREAD
1.1298424	1.3330715	41.3	-2.541997	6.5679770	5.612034	2.5005608	2.726713	1.485668	0.29
1.1488933	0.9780111	41.3	-5.526698	4.7519637	4.036060	-0.6641325	-5.659567	1.521604	0.23
1.1384379	0.9746862	41.3	-5.653060	2.5973912	6.463557	5.2299069	3.958412	3.361169	0.17
1.1647990	1.1298424	41.2	-2.273295	1.8443424	-4.095576	4.5418530	-8.413864	2.125417	0.11
1.2110008	1.1488933	41.3	-10.407194	1.1404043	-4.863195	3.8267196	1.839517	2.874708	0.11
1.3095620	1.1384379	41.2	-5.592855	0.4000961	-17.970490	-0.7398149	2.447104	-2.083045	0.17
6.8703760	-15.5411778	38.4	299.160275	-22.2665832	-65.170999	-9.0773697	-36.526831	-17.134668	0.61
5.8646967	-13.6362311	39.4	236.958845	-16.0975813	-44.854869	-6.7970189	-16.768648	-11.556949	0.62
2.8365572	-9.2007017	39.9	-57.591012	0.2856726	29.004894	-1.2746359	-7.501603	15.744309	0.65
2.0270341	6.8703760	40.7	-113.855563	22.2534510	81.950345	8.0480632	33.015374	14.958142	0.53
1.1986318	5.8646967	41.1	-83.543821	15.5077789	75.414169	8.8084739	19.376894	14.988075	0.55
0.5977556	2.8365572	41.1	-58.095308	3.6874164	6.520319	6.3229007	20.550075	8.067827	0.59

Table 2: Table 2 - Descriptive statistics of the series

	EMPL	HOURS	CLAIMS	ORDERSC	ORDERSI	ORDERSN	BUILDING	STOCK	SPREAD
Observations	669.0000	669.0000	669.0000	669.0000	669.0000	669.0000	669.0000	669.0000	669.0000
Minimum	-15.5412	37.3000	-113.8556	-22.2666	-70.0929	-27.0615	-46.0053	-37.2393	-6.5100
1st quartile	0.2433	40.4000	-5.2075	-1.2455	-6.4314	-2.2668	-4.8588	-1.8603	0.1200
Mean	0.3890	40.8130	0.6273	0.3424	0.0125	0.4293	0.0947	1.6449	1.0413
Median	0.4601	40.8000	-0.6356	0.3855	-0.7042	0.4911	1.1115	2.3280	1.2400
3rd quartile	0.7209	41.4000	4.2815	1.9869	6.2889	3.5989	5.9309	6.0512	2.2600
Maximum	6.8704	42.4000	299.1603	22.2535	81.9503	28.8569	46.2279	23.0685	3.8500
Desv. Pad.	1.1044	0.7463	20.5426	3.5239	13.8578	5.8918	10.6022	7.2105	1.6828

```
rbind(head(df), tail(df)) %>% kable("latex", caption = "Table 1 - First and last observations of the da
    kable_styling(bootstrap_options = c("striped",
        "hover"), font_size = 7)
```

```
descriptives <- matrix(NA, nrow = 8, ncol = (ncol(df) -
    1))
rownames(descriptives) <- c("Observations", "Minimum",
    "1st quartile", "Mean", "Median", "3rd quartile",
    "Maximum", "Desv. Pad.")

for (j in 1:ncol(df) - 1) {
    for (i in 1:8) {
        descriptives[i, j] <- round(as.numeric(desc(df[,
            j + 1])[i]), 4)
    }
}


descriptives <- data.frame(descriptives)
names(descriptives) <- colnames(df[, 2:ncol(df)])

descriptives %>% kable("latex", caption = "Table 2 - Descriptive statistics of the series") %>%
    kable_styling(bootstrap_options = c("striped",
        "hover"), font_size = 7)
```

The next does the graphs. Since they are in the original report, we are not plotting them again here.

```
tsDataEMP <- xts(df[, 2], order.by = dfData01[, 1],
    frequency = 12)

# Building the recessions Data from recessions
# available at:
# https://www.nber.org/research/business-cycle-dating
recessions_1 <- c("1969-12-01", "1973-11-01", "1980-01-01",
    "1981-07-01", "1990-07-01", "2001-03-01", "2007-12-01",
    "2020-02-01")

recessions_2 <- c("1970-11-01", "1975-03-01", "1980-07-01",
    "1982-11-01", "1991-03-01", "2001-11-01", "2009-06-01",
    "2020-09-01")

recessions.month <- data.frame(1:length(recessions_1),
    1:length(recessions_1), 1:length(recessions_1),
    as.Date(recessions_1), as.Date(recessions_2))
names(recessions.month) <- c("DATE", "Variable", "Value",
    "Peak", "Trough")


# Making the graph
plotSeries <- ggplot(dfData01[, c(1, 2)]) + geom_line(aes(x = as.Date(as.yearmon(Month)),
    y = EMP), alpha = 1, size = 0.1) + labs(title = "Quarterly  growth  rate of nonfarm payroll (EMP)",
    y = "EMP growth", x = "Date") + scale_x_date(date_breaks = "36 months",
    date_labels = "%m-%Y", limits = as.Date(c("1965-01-01",
        "2020-09-01"))) + theme_bw() + theme(axis.text.x = element_text(angle = 25,
    hjust = 1, size = 6), legend.position = "none") +
    geom_rect(data = recessions.month, aes(xmin = Peak,
        xmax = Trough, ymin = -Inf, ymax = +Inf), fill = "grey",
        alpha = 0.4)


p1 <- autoplot.acf01(ggPacf01(tsDataEMP, plot = FALSE,
    lag.max = 20, type = "correlation"))
p2 <- autoplot.acf01(ggPacf01(tsDataEMP, plot = FALSE,
    lag.max = 20, type = "partial"))

p3 <- grid.arrange(plotSeries, grid.arrange(p1, p2,
    nrow = 1), nrow = 2)


pdf(file = "C:\\Users\\aisha\\Dropbox\\ATE Assignment 3\\Fig_acf.pdf",
    width = 6, height = 6)
grid.arrange(p3)
dev.off()


cores <- brewer.pal(8, "Dark2")

# Plotting the other series
plotSeries01 <- ggplot(dfData01[, c(1, 4)]) + geom_line(aes(x = as.Date(as.yearmon(Month)),
    y = HOURS), alpha = 1, size = 0.5, colour = cores[1]) +
```

```
    labs(title = "Quarterly average weekly hours, manufacturing",
        y = "Hours", x = "Date") + scale_x_date(date_breaks = "48 months",
    date_labels = "%m-%Y", limits = as.Date(c("1965-01-01",
        "2020-09-01"))) + theme_bw() + theme(axis.text.x = element_text(angle = 25,
    hjust = 1, size = 6), legend.position = "none",
    plot.title = element_text(size = 9), axis.title.y = element_text(size = 7),
    axis.title.x = element_text(size = 7)) + geom_rect(data = recessions.month,
    aes(xmin = Peak, xmax = Trough, ymin = -Inf, ymax = +Inf),
    fill = "grey", alpha = 0.4)

plotSeries02 <- ggplot(dfData01[, c(1, 5)]) + geom_line(aes(x = as.Date(as.yearmon(Month)),
    y = CLAIMS), alpha = 1, size = 0.5, colour = cores[2]) +
    labs(title = "Quarterly average growth claims for UI",
        y = "UI Claims", x = "Date") + scale_x_date(date_breaks = "48 months",
    date_labels = "%m-%Y", limits = as.Date(c("1965-01-01",
        "2020-09-01"))) + theme_bw() + theme(axis.text.x = element_text(angle = 25,
    hjust = 1, size = 6), legend.position = "none",
    plot.title = element_text(size = 9), axis.title.y = element_text(size = 7),
    axis.title.x = element_text(size = 7)) + geom_rect(data = recessions.month,
    aes(xmin = Peak, xmax = Trough, ymin = -Inf, ymax = +Inf),
    fill = "grey", alpha = 0.4)

plotSeries03 <- ggplot(dfData01[, c(1, 6)]) + geom_line(aes(x = as.Date(as.yearmon(Month)),
    y = ORDERSC), alpha = 1, size = 0.5, colour = cores[3]) +
    labs(title = "Quarterly growth of consumer goods orders",
        y = "New orders", x = "Date") + scale_x_date(date_breaks = "48 months",
    date_labels = "%m-%Y", limits = as.Date(c("1965-01-01",
        "2020-09-01"))) + theme_bw() + theme(axis.text.x = element_text(angle = 25,
    hjust = 1, size = 6), legend.position = "none",
    plot.title = element_text(size = 9), axis.title.y = element_text(size = 7),
    axis.title.x = element_text(size = 7)) + geom_rect(data = recessions.month,
    aes(xmin = Peak, xmax = Trough, ymin = -Inf, ymax = +Inf),
    fill = "grey", alpha = 0.4)


plotSeries04 <- ggplot(dfData01[, c(1, 7)]) + geom_line(aes(x = as.Date(as.yearmon(Month)),
    y = ORDERSI), alpha = 1, size = 0.5, colour = cores[4]) +
    labs(title = "Quarterly growth of ISM new orders index",
        y = "ISM index", x = "Date") + scale_x_date(date_breaks = "48 months",
    date_labels = "%m-%Y", limits = as.Date(c("1965-01-01",
        "2020-09-01"))) + theme_bw() + theme(axis.text.x = element_text(angle = 25,
    hjust = 1, size = 6), legend.position = "none",
    plot.title = element_text(size = 9), axis.title.y = element_text(size = 7),
    axis.title.x = element_text(size = 7)) + geom_rect(data = recessions.month,
    aes(xmin = Peak, xmax = Trough, ymin = -Inf, ymax = +Inf),
    fill = "grey", alpha = 0.4)

plotSeries05 <- ggplot(dfData01[, c(1, 8)]) + geom_line(aes(x = as.Date(as.yearmon(Month)),
    y = ORDERSN), alpha = 1, size = 0.5, colour = cores[5]) +
    labs(title = "Quarterly growth of capital goods orders",
        y = "New orders", x = "Date") + scale_x_date(date_breaks = "48 months",
    date_labels = "%m-%Y", limits = as.Date(c("1965-01-01",
        "2020-09-01"))) + theme_bw() + theme(axis.text.x = element_text(angle = 25,
```

```
    hjust = 1, size = 6), legend.position = "none",
    plot.title = element_text(size = 9), axis.title.y = element_text(size = 7),
    axis.title.x = element_text(size = 7)) + geom_rect(data = recessions.month,
    aes(xmin = Peak, xmax = Trough, ymin = -Inf, ymax = +Inf),
    fill = "grey", alpha = 0.4)

plotSeries06 <- ggplot(dfData01[, c(1, 9)]) + geom_line(aes(x = as.Date(as.yearmon(Month)),
    y = BUILDING), alpha = 1, size = 0.5, colour = cores[6]) +
    labs(title = "Quarterly growth of building permits",
        y = "Building permits", x = "Date") + scale_x_date(date_breaks = "48 months",
    date_labels = "%m-%Y", limits = as.Date(c("1965-01-01",
        "2020-09-01"))) + theme_bw() + theme(axis.text.x = element_text(angle = 25,
    hjust = 1, size = 6), legend.position = "none",
    plot.title = element_text(size = 9), axis.title.y = element_text(size = 7),
    axis.title.x = element_text(size = 7)) + geom_rect(data = recessions.month,
    aes(xmin = Peak, xmax = Trough, ymin = -Inf, ymax = +Inf),
    fill = "grey", alpha = 0.4)

plotSeries07 <- ggplot(dfData01[, c(1, 10)]) + geom_line(aes(x = as.Date(as.yearmon(Month)),
    y = STOCK), alpha = 1, size = 0.5, colour = cores[7]) +
    labs(title = "Quarterly growth of stock prices index",
        y = "S&P 500s", x = "Date") + scale_x_date(date_breaks = "48 months",
    date_labels = "%m-%Y", limits = as.Date(c("1965-01-01",
        "2020-09-01"))) + theme_bw() + theme(axis.text.x = element_text(angle = 25,
    hjust = 1, size = 6), legend.position = "none",
    plot.title = element_text(size = 9), axis.title.y = element_text(size = 7),
    axis.title.x = element_text(size = 7)) + geom_rect(data = recessions.month,
    aes(xmin = Peak, xmax = Trough, ymin = -Inf, ymax = +Inf),
    fill = "grey", alpha = 0.4)

plotSeries08 <- ggplot(dfData01[, c(1, 11)]) + geom_line(aes(x = as.Date(as.yearmon(Month)),
    y = SPREAD), alpha = 1, size = 0.5, colour = cores[8]) +
    labs(title = "Interest rate spread", y = "S&P 500s",
        x = "Date") + scale_x_date(date_breaks = "48 months",
    date_labels = "%m-%Y", limits = as.Date(c("1965-01-01",
        "2020-09-01"))) + theme_bw() + theme(axis.text.x = element_text(angle = 25,
    hjust = 1, size = 6), legend.position = "none",
    plot.title = element_text(size = 9), axis.title.y = element_text(size = 7),
    axis.title.x = element_text(size = 7)) + geom_rect(data = recessions.month,
    aes(xmin = Peak, xmax = Trough, ymin = -Inf, ymax = +Inf),
    fill = "grey", alpha = 0.4)


pdf(file = "C:\\Users\\aisha\\Dropbox\\ATE Assignment 3\\Fig_covariates.pdf",
    width = 7, height = 8)
grid.arrange(plotSeries01, plotSeries02, plotSeries03,
    plotSeries04, plotSeries05, plotSeries06, plotSeries07,
    plotSeries08, ncol = 2)
dev.off()
```

Forecast results

For the forecast results we assemble a new data frame with the forecasts and the actual values and do plots, compute forecast errors and compute the DM test.

```
# Merge all forecasts
result <- cbind(tail(dfData01[, 2], nrow(pred1)), pred1,
    pred2, pred3, pred4, pred5, pred6, pred7)
result <- as.xts(result, order.by = tail(dfData01[,
    1], nrow(pred1)))
names(result)[1] <- "y"
names(result)[2] <- "lin.reg"
names(result)[3] <- "avlin.reg"
names(result)[4] <- "fac.aug"
names(result)[5] <- "pe.Lasso"
names(result)[6] <- "ll.forest"
names(result)[7] <- "ar1"
names(result)[8] <- "hist.avg"
result2 <- data.frame(as.yearmon(tail(dfData01[, 1],
    nrow(pred1))), result)
names(result2)[1] <- "date"

# save(result, file = 'result.RData')

# Plot all forecasts removing 2020
p1 <- ggplot(head(result2, n = -9), aes(as.Date(date),
    y)) + geom_line(aes(colour = "skyblue4")) + geom_line(data = head(result2,
    n = -9), aes(x = as.Date(date), y = lin.reg, colour = "red3"),
    alpha = 0.8, linetype = "dashed", size = 0.6) +
    scale_x_date(labels = date_format("%m-%Y"), date_breaks = "36 months",
        minor_breaks = NULL, limits = c(as.Date("1980-01-01"),
            as.Date("2019-12-01"))) + theme_bw() +
    scale_colour_manual(values = c("red3", "skyblue4"),
        name = "Series", labels = c("Forecast", "EMP")) +
    labs(title = "Linear regression", x = "Date", y = "EMP")

p1 <- p1 + theme(axis.text.x = element_text(angle = 25,
    hjust = 1, size = 6), axis.title.x = element_blank(),
    axis.title.y = element_text(size = 7), plot.title = element_text(size = 9),
    legend.title = element_text(size = 8), legend.text = element_text(size = 7),
    legend.position = "bottom")


p2 <- ggplot(head(result2, n = -9), aes(as.Date(date),
    y)) + geom_line(aes(colour = "skyblue4")) + geom_line(data = head(result2,
    n = -9), aes(x = as.Date(date), y = avlin.reg,
    colour = "red3"), alpha = 0.8, linetype = "dashed",
    size = 0.6) + scale_x_date(labels = date_format("%m-%Y"),
    date_breaks = "36 months", minor_breaks = NULL,
    limits = c(as.Date("1980-01-01"), as.Date("2019-12-01"))) +
    theme_bw() + scale_colour_manual(values = c("red3",
    "skyblue4"), name = "Series", labels = c("Forecast",
    "EMP")) + labs(title = "Weighted average linear regressions",
    x = "Date", y = "EMP")
```

```
p2 <- p2 + theme(axis.text.x = element_text(angle = 25,
    hjust = 1, size = 6), axis.title.x = element_blank(),
    axis.title.y = element_text(size = 7), plot.title = element_text(size = 9),
    legend.title = element_text(size = 8), legend.text = element_text(size = 7),
    legend.position = "bottom")


p3 <- ggplot(head(result2, n = -9), aes(as.Date(date),
    y)) + geom_line(aes(colour = "skyblue4")) + geom_line(data = head(result2,
    n = -9), aes(x = as.Date(date), y = fac.aug, colour = "red3"),
    alpha = 0.8, linetype = "dashed", size = 0.6) +
    scale_x_date(labels = date_format("%m-%Y"), date_breaks = "36 months",
        minor_breaks = NULL, limits = c(as.Date("1980-01-01"),
            as.Date("2019-12-01"))) + theme_bw() +
    scale_colour_manual(values = c("red3", "skyblue4"),
        name = "Series", labels = c("Forecast", "EMP")) +
    labs(title = "Factor augmented", x = "Date", y = "EMP")

p3 <- p3 + theme(axis.text.x = element_text(angle = 25,
    hjust = 1, size = 6), axis.title.x = element_blank(),
    axis.title.y = element_text(size = 7), plot.title = element_text(size = 9),
    legend.title = element_text(size = 8), legend.text = element_text(size = 7),
    legend.position = "bottom")


p4 <- ggplot(head(result2, n = -9), aes(as.Date(date),
    y)) + geom_line(aes(colour = "skyblue4")) + geom_line(data = head(result2,
    n = -9), aes(x = as.Date(date), y = pe.Lasso, colour = "red3"),
    alpha = 0.8, linetype = "dashed", size = 0.6) +
    scale_x_date(labels = date_format("%m-%Y"), date_breaks = "36 months",
        minor_breaks = NULL, limits = c(as.Date("1980-01-01"),
            as.Date("2019-12-01"))) + theme_bw() +
    scale_colour_manual(values = c("red3", "skyblue4"),
        name = "Series", labels = c("Forecast", "EMP")) +
    labs(title = "Pe-LASSO", x = "Date", y = "EMP")

p4 <- p4 + theme(axis.text.x = element_text(angle = 25,
    hjust = 1, size = 6), axis.title.x = element_blank(),
    axis.title.y = element_text(size = 7), plot.title = element_text(size = 9),
    legend.title = element_text(size = 8), legend.text = element_text(size = 7),
    legend.position = "bottom")


p5 <- ggplot(head(result2, n = -9), aes(as.Date(date),
    y)) + geom_line(aes(colour = "skyblue4")) + geom_line(data = head(result2,
    n = -9), aes(x = as.Date(date), y = ll.forest,
    colour = "red3"), alpha = 0.8, linetype = "dashed",
    size = 0.6) + scale_x_date(labels = date_format("%m-%Y"),
    date_breaks = "36 months", minor_breaks = NULL,
    limits = c(as.Date("1980-01-01"), as.Date("2019-12-01"))) +
    theme_bw() + scale_colour_manual(values = c("red3",
    "skyblue4"), name = "Series", labels = c("Forecast",
    "EMP")) + labs(title = "Local linear forest", x = "Date",
```

```
    y = "EMP")

p5 <- p5 + theme(axis.text.x = element_text(angle = 25,
    hjust = 1, size = 6), axis.title.x = element_blank(),
    axis.title.y = element_text(size = 7), plot.title = element_text(size = 9),
    legend.title = element_text(size = 8), legend.text = element_text(size = 7),
    legend.position = "bottom")


p6 <- ggplot(head(result2, n = -9), aes(as.Date(date),
    y)) + geom_line(aes(colour = "skyblue4")) + geom_line(data = head(result2,
    n = -9), aes(x = as.Date(date), y = ar1, colour = "red3"),
    alpha = 0.8, linetype = "dashed", size = 0.6) +
    geom_line(data = head(result2, n = -9), aes(x = as.Date(date),
        y = hist.avg, colour = "darkorchid3"), alpha = 0.8) +
    scale_x_date(labels = date_format("%m-%Y"), date_breaks = "36 months",
        minor_breaks = NULL, limits = c(as.Date("1980-01-01"),
            as.Date("2019-12-01"))) + theme_bw() +
    scale_colour_manual(values = c("darkorchid3", "red3",
        "skyblue4"), name = "Series", labels = c("Hist. avg",
        "AR(1)", "EMP")) + labs(title = "Benchmark forecasts",
    x = "Date", y = "EMP")

p6 <- p6 + theme(axis.text.x = element_text(angle = 25,
    hjust = 1, size = 6), axis.title.x = element_blank(),
    axis.title.y = element_text(size = 7), plot.title = element_text(size = 9),
    legend.title = element_text(size = 8), legend.text = element_text(size = 7),
    legend.position = "bottom")


pdf(file = "C:\\Users\\aisha\\Dropbox\\ATE Assignment 3\\Fig_forecasts.pdf",
    width = 7, height = 8)
grid.arrange(p6, p1, p2, p3, p4, p5, ncol = 2)
dev.off()


# Plot all forecasts for the year of 2020

p1 <- ggplot(tail(result2, n = 15), aes(as.Date(date),
    y)) + geom_line(aes(colour = "skyblue4")) + geom_line(data = tail(result2,
    n = 15), aes(x = as.Date(date), y = lin.reg, colour = "red3"),
    alpha = 0.8, linetype = "dashed", size = 0.6) +
    scale_x_date(labels = date_format("%m-%Y"), date_breaks = "2 months",
        minor_breaks = NULL, limits = c(as.Date("2019-07-01"),
            as.Date("2020-09-01"))) + theme_bw() +
    scale_colour_manual(values = c("red3", "skyblue4"),
        name = "Series", labels = c("Forecast", "EMP")) +
    labs(title = "Linear regression", x = "Date", y = "EMP")

p1 <- p1 + theme(axis.text.x = element_text(angle = 25,
    hjust = 1, size = 6), axis.title.x = element_blank(),
    axis.title.y = element_text(size = 7), plot.title = element_text(size = 9),
    legend.title = element_text(size = 8), legend.text = element_text(size = 7),
```

```
    legend.position = "bottom")


p2 <- ggplot(tail(result2, n = 15), aes(as.Date(date),
    y)) + geom_line(aes(colour = "skyblue4")) + geom_line(data = tail(result2,
    n = 15), aes(x = as.Date(date), y = avlin.reg,
    colour = "red3"), alpha = 0.8, linetype = "dashed",
    size = 0.6) + scale_x_date(labels = date_format("%m-%Y"),
    date_breaks = "2 months", minor_breaks = NULL,
    limits = c(as.Date("2019-07-01"), as.Date("2020-09-01"))) +
    theme_bw() + scale_colour_manual(values = c("red3",
    "skyblue4"), name = "Series", labels = c("Forecast",
    "EMP")) + labs(title = "Weighted average linear regressions",
    x = "Date", y = "EMP")

p2 <- p2 + theme(axis.text.x = element_text(angle = 25,
    hjust = 1, size = 6), axis.title.x = element_blank(),
    axis.title.y = element_text(size = 7), plot.title = element_text(size = 9),
    legend.title = element_text(size = 8), legend.text = element_text(size = 7),
    legend.position = "bottom")


p3 <- ggplot(tail(result2, n = 15), aes(as.Date(date),
    y)) + geom_line(aes(colour = "skyblue4")) + geom_line(data = tail(result2,
    n = 15), aes(x = as.Date(date), y = fac.aug, colour = "red3"),
    alpha = 0.8, linetype = "dashed", size = 0.6) +
    scale_x_date(labels = date_format("%m-%Y"), date_breaks = "2 months",
        minor_breaks = NULL, limits = c(as.Date("2019-07-01"),
            as.Date("2020-09-01"))) + theme_bw() +
    scale_colour_manual(values = c("red3", "skyblue4"),
        name = "Series", labels = c("Forecast", "EMP")) +
    labs(title = "Factor augmented", x = "Date", y = "EMP")

p3 <- p3 + theme(axis.text.x = element_text(angle = 25,
    hjust = 1, size = 6), axis.title.x = element_blank(),
    axis.title.y = element_text(size = 7), plot.title = element_text(size = 9),
    legend.title = element_text(size = 8), legend.text = element_text(size = 7),
    legend.position = "bottom")


p4 <- ggplot(tail(result2, n = 15), aes(as.Date(date),
    y)) + geom_line(aes(colour = "skyblue4")) + geom_line(data = tail(result2,
    n = 15), aes(x = as.Date(date), y = pe.Lasso, colour = "red3"),
    alpha = 0.8, linetype = "dashed", size = 0.6) +
    scale_x_date(labels = date_format("%m-%Y"), date_breaks = "2 months",
        minor_breaks = NULL, limits = c(as.Date("2019-07-01"),
            as.Date("2020-09-01"))) + theme_bw() +
    scale_colour_manual(values = c("red3", "skyblue4"),
        name = "Series", labels = c("Forecast", "EMP")) +
    labs(title = "Pe-LASSO", x = "Date", y = "EMP")

p4 <- p4 + theme(axis.text.x = element_text(angle = 25,
    hjust = 1, size = 6), axis.title.x = element_blank(),
```

```
    axis.title.y = element_text(size = 7), plot.title = element_text(size = 9),
    legend.title = element_text(size = 8), legend.text = element_text(size = 7),
    legend.position = "bottom")


p5 <- ggplot(tail(result2, n = 15), aes(as.Date(date),
    y)) + geom_line(aes(colour = "skyblue4")) + geom_line(data = tail(result2,
    n = 15), aes(x = as.Date(date), y = ll.forest,
    colour = "red3"), alpha = 0.8, linetype = "dashed",
    size = 0.6) + scale_x_date(labels = date_format("%m-%Y"),
    date_breaks = "2 months", minor_breaks = NULL,
    limits = c(as.Date("2019-07-01"), as.Date("2020-09-01"))) +
    theme_bw() + scale_colour_manual(values = c("red3",
    "skyblue4"), name = "Series", labels = c("Forecast",
    "EMP")) + labs(title = "Local linear forest", x = "Date",
    y = "EMP")

p5 <- p5 + theme(axis.text.x = element_text(angle = 25,
    hjust = 1, size = 6), axis.title.x = element_blank(),
    axis.title.y = element_text(size = 7), plot.title = element_text(size = 9),
    legend.title = element_text(size = 8), legend.text = element_text(size = 7),
    legend.position = "bottom")


p6 <- ggplot(tail(result2, n = 15), aes(as.Date(date),
    y)) + geom_line(aes(colour = "skyblue4")) + geom_line(data = tail(result2,
    n = 15), aes(x = as.Date(date), y = ar1, colour = "red3"),
    alpha = 0.8, linetype = "dashed", size = 0.6) +
    geom_line(data = tail(result2, n = 15), aes(x = as.Date(date),
        y = hist.avg, colour = "darkorchid3"), alpha = 0.8) +
    scale_x_date(labels = date_format("%m-%Y"), date_breaks = "2 months",
        minor_breaks = NULL, limits = c(as.Date("2019-07-01"),
            as.Date("2020-09-01"))) + theme_bw() +
    scale_colour_manual(values = c("darkorchid3", "red3",
        "skyblue4"), name = "Series", labels = c("Hist. avg",
        "AR(1)", "EMP")) + labs(title = "Benchmark forecasts",
    x = "Date", y = "EMP")

p6 <- p6 + theme(axis.text.x = element_text(angle = 25,
    hjust = 1, size = 6), axis.title.x = element_blank(),
    axis.title.y = element_text(size = 7), plot.title = element_text(size = 9),
    legend.title = element_text(size = 8), legend.text = element_text(size = 7),
    legend.position = "bottom")


pdf(file = "C:\\Users\\aisha\\Dropbox\\ATE Assignment 3\\Fig_2020forecast.pdf",
    width = 7, height = 8)
grid.arrange(p6, p1, p2, p3, p4, p5, ncol = 2)
dev.off()


# Compute mse, rmse, mae
MSE <- rep(NA, ncol(result) - 1)
```

```
RMSE <- rep(NA, ncol(result) - 1)
MAE <- rep(NA, ncol(result) - 1)

for (i in 2:ncol(result)) {
    MSE[i - 1] <- mse(result[, 1], result[, i])
    RMSE[i - 1] <- rmse(result[, 1], result[, i])
    MAE[i - 1] <- mae(result[, 1], result[, i])
}

methods_names <- tail(names(result), n = 7)

results_metrics <- matrix(NA, nrow = 3, ncol = length(methods_names))
rownames(results_metrics) <- c("MSE", "RMSE", "MAE")

results_metrics[1, ] <- round(MSE, 4)
results_metrics[2, ] <- round(RMSE, 4)
results_metrics[3, ] <- round(MAE, 4)


results_metrics <- data.frame(results_metrics)
names(results_metrics) <- methods_names

results_metrics %>% kable("latex") %>% kable_styling(bootstrap_options = c("striped",
    "hover"))

# Run DM test
DM_results <- matrix(NA, length(methods_names), ncol = length(methods_names))


# For the linear regression
DM_results[1, ] <- round(c(DM.test(f1 = result$lin.reg,
    f2 = result$lin.reg, y = result$y, c = TRUE)$p.value,
    DM.test(f1 = result$lin.reg, f2 = result$avlin.reg,
        y = result$y, c = TRUE)$p.value, DM.test(f1 = result$lin.reg,
        f2 = result$fac.aug, y = result$y, c = TRUE)$p.value,
    DM.test(f1 = result$lin.reg, f2 = result$pe.Lasso,
        y = result$y, c = TRUE)$p.value, DM.test(f1 = result$lin.reg,
        f2 = result$ll.forest, y = result$y, c = TRUE)$p.value,
    DM.test(f1 = result$lin.reg, f2 = result$ar1, y = result$y,
        c = TRUE)$p.value, DM.test(f1 = result$lin.reg,
        f2 = result$hist.avg, y = result$y, c = TRUE)$p.value),
    4)

# For the linear regression
DM_results[2, ] <- round(c(DM.test(f1 = result$avlin.reg,
    f2 = result$lin.reg, y = result$y, c = TRUE)$p.value,
    DM.test(f1 = result$avlin.reg, f2 = result$avlin.reg,
        y = result$y, c = TRUE)$p.value, DM.test(f1 = result$avlin.reg,
        f2 = result$fac.aug, y = result$y, c = TRUE)$p.value,
    DM.test(f1 = result$avlin.reg, f2 = result$pe.Lasso,
        y = result$y, c = TRUE)$p.value, DM.test(f1 = result$avlin.reg,
        f2 = result$ll.forest, y = result$y, c = TRUE)$p.value,
    DM.test(f1 = result$avlin.reg, f2 = result$ar1,
```

```
        y = result$y, c = TRUE)$p.value, DM.test(f1 = result$avlin.reg,
        f2 = result$hist.avg, y = result$y, c = TRUE)$p.value),
    4)


# For the factor augmented model
DM_results[3, ] <- round(c(DM.test(f1 = result$fac.aug,
    f2 = result$lin.reg, y = result$y, c = TRUE)$p.value,
    DM.test(f1 = result$fac.aug, f2 = result$avlin.reg,
        y = result$y, c = TRUE)$p.value, DM.test(f1 = result$fac.aug,
        f2 = result$fac.aug, y = result$y, c = TRUE)$p.value,
    DM.test(f1 = result$fac.aug, f2 = result$pe.Lasso,
        y = result$y, c = TRUE)$p.value, DM.test(f1 = result$fac.aug,
        f2 = result$ll.forest, y = result$y, c = TRUE)$p.value,
    DM.test(f1 = result$fac.aug, f2 = result$ar1, y = result$y,
        c = TRUE)$p.value, DM.test(f1 = result$fac.aug,
        f2 = result$hist.avg, y = result$y, c = TRUE)$p.value),
    4)

# For the PeLASSO
DM_results[4, ] <- round(c(DM.test(f1 = result$pe.Lasso,
    f2 = result$lin.reg, y = result$y, c = TRUE)$p.value,
    DM.test(f1 = result$pe.Lasso, f2 = result$avlin.reg,
        y = result$y, c = TRUE)$p.value, DM.test(f1 = result$pe.Lasso,
        f2 = result$fac.aug, y = result$y, c = TRUE)$p.value,
    DM.test(f1 = result$pe.Lasso, f2 = result$pe.Lasso,
        y = result$y, c = TRUE)$p.value, DM.test(f1 = result$pe.Lasso,
        f2 = result$ll.forest, y = result$y, c = TRUE)$p.value,
    DM.test(f1 = result$pe.Lasso, f2 = result$ar1,
        y = result$y, c = TRUE)$p.value, DM.test(f1 = result$pe.Lasso,
        f2 = result$hist.avg, y = result$y, c = TRUE)$p.value),
    4)

# For the local linear forest
DM_results[5, ] <- round(c(DM.test(f1 = result$ll.forest,
    f2 = result$lin.reg, y = result$y, c = TRUE)$p.value,
    DM.test(f1 = result$ll.forest, f2 = result$avlin.reg,
        y = result$y, c = TRUE)$p.value, DM.test(f1 = result$ll.forest,
        f2 = result$fac.aug, y = result$y, c = TRUE)$p.value,
    DM.test(f1 = result$ll.forest, f2 = result$pe.Lasso,
        y = result$y, c = TRUE)$p.value, DM.test(f1 = result$ll.forest,
        f2 = result$ll.forest, y = result$y, c = TRUE)$p.value,
    DM.test(f1 = result$ll.forest, f2 = result$ar1,
        y = result$y, c = TRUE)$p.value, DM.test(f1 = result$ll.forest,
        f2 = result$hist.avg, y = result$y, c = TRUE)$p.value),
    4)

# For the AR1
DM_results[6, ] <- round(c(DM.test(f1 = result$ar1,
    f2 = result$lin.reg, y = result$y, c = TRUE)$p.value,
    DM.test(f1 = result$ar1, f2 = result$avlin.reg,
        y = result$y, c = TRUE)$p.value, DM.test(f1 = result$ar1,
        f2 = result$fac.aug, y = result$y, c = TRUE)$p.value,
```

```
    DM.test(f1 = result$ar1, f2 = result$pe.Lasso,
        y = result$y, c = TRUE)$p.value, DM.test(f1 = result$ar1,
        f2 = result$ll.forest, y = result$y, c = TRUE)$p.value,
    DM.test(f1 = result$ar1, f2 = result$ar1, y = result$y,
        c = TRUE)$p.value, DM.test(f1 = result$ar1,
        f2 = result$hist.avg, y = result$y, c = TRUE)$p.value),
    4)


DM_resultsdf <- data.frame(DM_results)
diag(DM_resultsdf) <- 1
names(DM_resultsdf) <- methods_names
row.names(DM_resultsdf) <- methods_names

DM_resultsdf %>% kable("latex") %>% kable_styling(bootstrap_options = c("striped",
    "hover"))

# Compute mse, rmse, mae - without 2020 included
result <- head(result, n = -9)

MSE <- rep(NA, ncol(result) - 1)
RMSE <- rep(NA, ncol(result) - 1)
MAE <- rep(NA, ncol(result) - 1)

for (i in 2:ncol(result)) {
    MSE[i - 1] <- mse(result[, 1], result[, i])
    RMSE[i - 1] <- rmse(result[, 1], result[, i])
    MAE[i - 1] <- mae(result[, 1], result[, i])
}

methods_names <- tail(names(result), n = 7)

results_metrics <- matrix(NA, nrow = 3, ncol = length(methods_names))
rownames(results_metrics) <- c("MSE", "RMSE", "MAE")

results_metrics[1, ] <- round(MSE, 4)
results_metrics[2, ] <- round(RMSE, 4)
results_metrics[3, ] <- round(MAE, 4)


results_metrics <- data.frame(results_metrics)
names(results_metrics) <- methods_names

results_metrics %>% kable("html") %>% kable_styling(bootstrap_options = c("striped",
    "hover"))
```