001inclass
ADVANCED METHODS III 140.753

```
## Load data
load("trainData.rda")

## Initial exploration
length(trainData)

## [1] 50

trainData[[1]][, 1:10]

##         [,1]   [,2]   [,3]   [,4]  [,5]   [,6]   [,7]   [,8]    [,9]   [,10]
## [1,] 0.36036 0.6607 0.6807 2.1421 2.222 2.5425 2.7427 3.0230  4.3644  6.2462
## [2,] 0.02966 0.3240 0.3429 0.9712 0.949 0.8015 0.6665 0.4343 -0.7784 -0.3594

trainData[[2]][, 1:10]

##         [,1]    [,2]    [,3]    [,4]    [,5]    [,6]    [,7]    [,8]    [,9]   [,10]
## [1,]  0.3203  0.4204  0.4404  0.6406  0.6807  0.9810  1.7017  1.8218  2.1021  2.2823
## [2,] -0.1544 -0.2523 -0.2716 -0.4576 -0.4928 -0.7282 -0.9994 -0.9963 -0.9338 -0.8545


par(mfrow = c(1, 2))
plot(trainData[[1]][1, ], trainData[[1]][2, ])
plot(trainData[[1]][1, ])
```

The goal is to predict variable 1 at time points 18, 19 and 20.

```
### Summarize information according to the method I submitted
mat <- matrix(NA, ncol = 50, nrow = ncol(trainData[[1]]))
for (k in 1:ncol(trainData[[1]])) {
    for (i in 1:length(trainData)) {
        if (k == 1) {
            cols <- which(trainData[[i]][1, ] <= trainData[[1]][1, k])
        } else {
            cols <- which(trainData[[i]][1, ] <= trainData[[1]][1, k] & trainData[[i]][1, ] >
                trainData[[1]][1, k - 1])
        }
        if (length(cols) > 0) {
            mat[k, i] <- mean(trainData[[i]][2, cols])
        } else {
            mat[k, i] <- NA
        }
    }
}
```

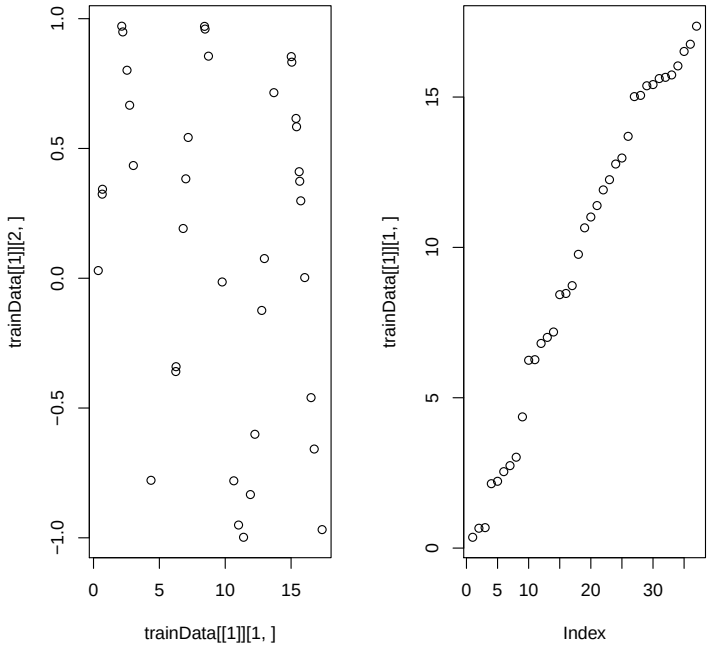


Figure 1: Initial exploration of the data

```
## Explore summary of info summary(mat) ## Quite big, so I'm just going to show the first
## 4
summary(mat[, 1:4])
```

```
##        V1                 V2                 V3                 V4
##  Min.   :-0.998   Min.   :-0.976   Min.   :-0.993   Min.   :-0.984
##  1st Qu.:-0.359   1st Qu.:-0.615   1st Qu.:-0.757   1st Qu.:-0.699
##  Median : 0.324   Median :-0.169   Median : 0.068   Median :-0.136
##  Mean   : 0.144   Mean   :-0.095   Mean   :-0.084   Mean   :-0.062
##  3rd Qu.: 0.666   3rd Qu.: 0.554   3rd Qu.: 0.339   3rd Qu.: 0.522
##  Max.   : 0.971   Max.   : 0.994   Max.   : 0.996   Max.   : 0.997
##                   NA's   :14       NA's   :19       NA's   :16
```

```
heatmap(mat)
```

```
## explore V1
qqnorm(trainData[[1]][2, ])
qqline(trainData[[1]][2, ])
```

From figure 3 it seems that it is reasonable to claim that it follows a normal distribution and thus could use `lm`.

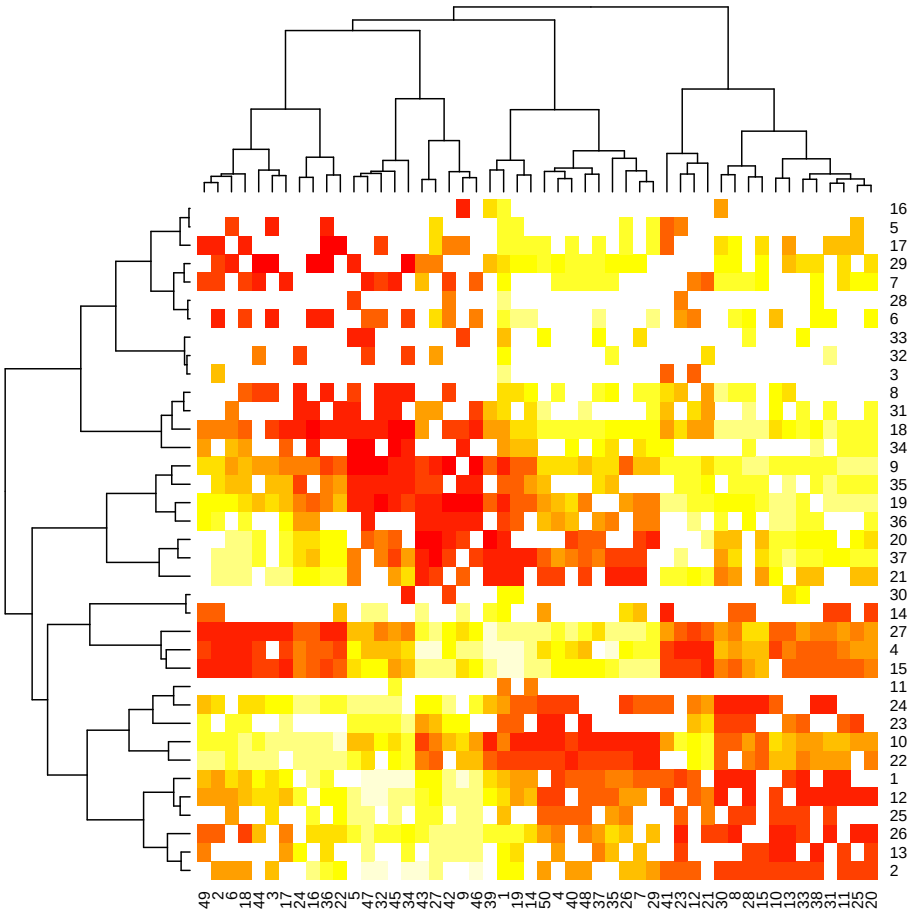


Figure 2: Simple heatmap of the summarized data. Created with default params.

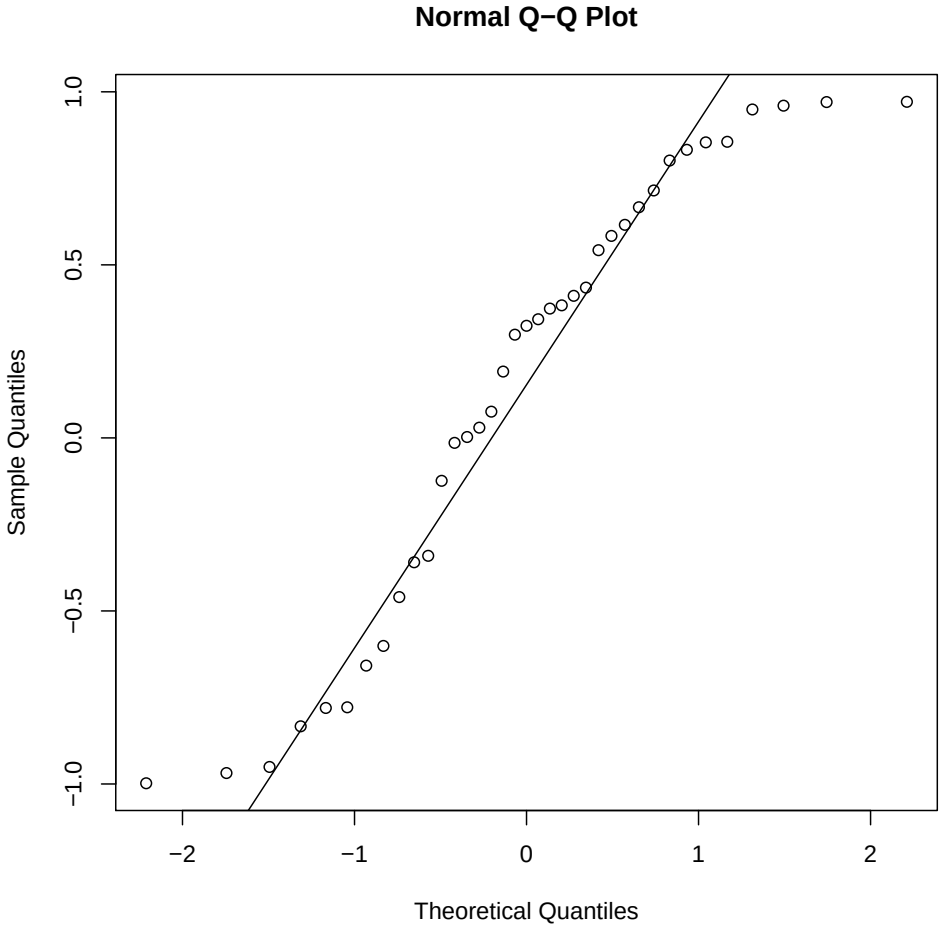


Figure 3: Exploring V1

```
## Impute using column means.
mat2 <- mat
for (i in 1:ncol(mat)) {
    mat2[is.na(mat[, i]), i] <- mean(mat[, i], na.rm = TRUE)
}

## Then fit a lm
colnames(mat2) <- paste("V", 1:ncol(mat2), sep = "")
mat2 <- data.frame(mat2)
fit <- lm(V1 ~ ., data = mat2[1:17, ])
summary(fit)
```

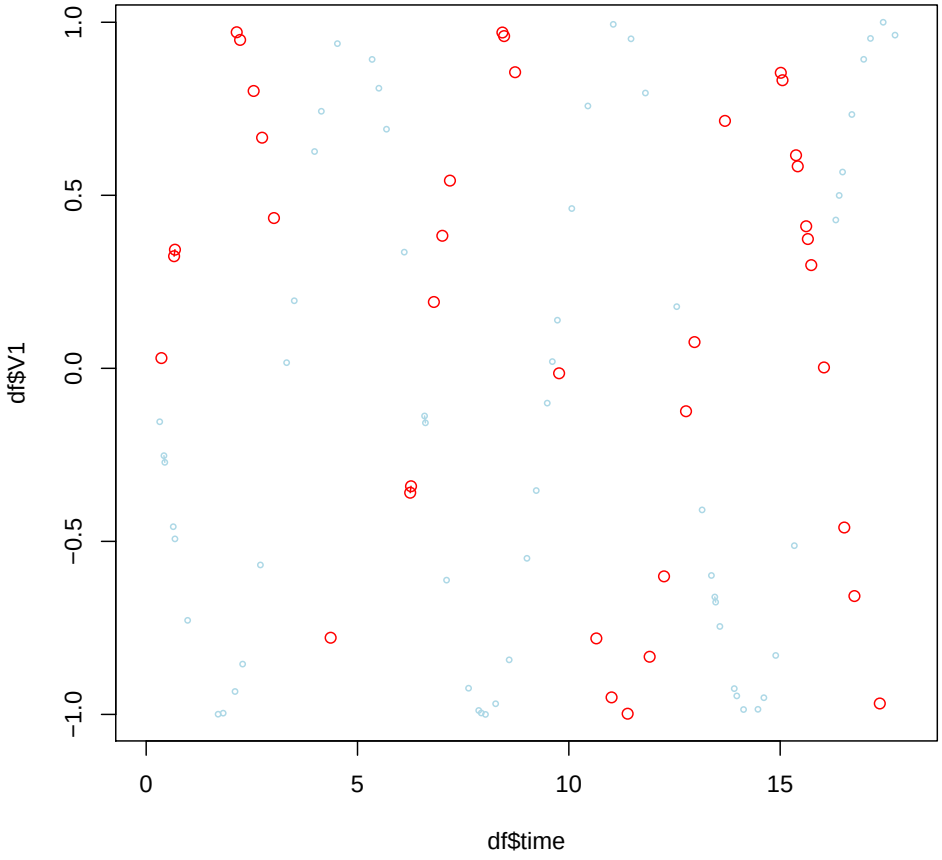
```
##
## Call:
## lm(formula = V1 ~ ., data = mat2[1:17, ])
##
## Residuals:
## ALL 17 residuals are 0: no residual degrees of freedom!
##
## Coefficients: (33 not defined because of singularities)
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   0.0774         NA      NA       NA
## V2            2.6474         NA      NA       NA
## V3           -8.6855         NA      NA       NA
## V4            1.5932         NA      NA       NA
## V5            2.7793         NA      NA       NA
## V6            9.0149         NA      NA       NA
## V7            5.1101         NA      NA       NA
## V8           -7.0987         NA      NA       NA
## V9            4.8354         NA      NA       NA
## V10          -8.1568         NA      NA       NA
## V11          10.3656         NA      NA       NA
## V12           0.0760         NA      NA       NA
## V13           0.3256         NA      NA       NA
## V14           3.9013         NA      NA       NA
## V15           2.7922         NA      NA       NA
## V16           2.9768         NA      NA       NA
## V17           2.9693         NA      NA       NA
## V18               NA         NA      NA       NA
## V19               NA         NA      NA       NA
## V20               NA         NA      NA       NA
## V21               NA         NA      NA       NA
## V22               NA         NA      NA       NA
## V23               NA         NA      NA       NA
## V24               NA         NA      NA       NA
## V25               NA         NA      NA       NA
## V26               NA         NA      NA       NA
## V27               NA         NA      NA       NA
```

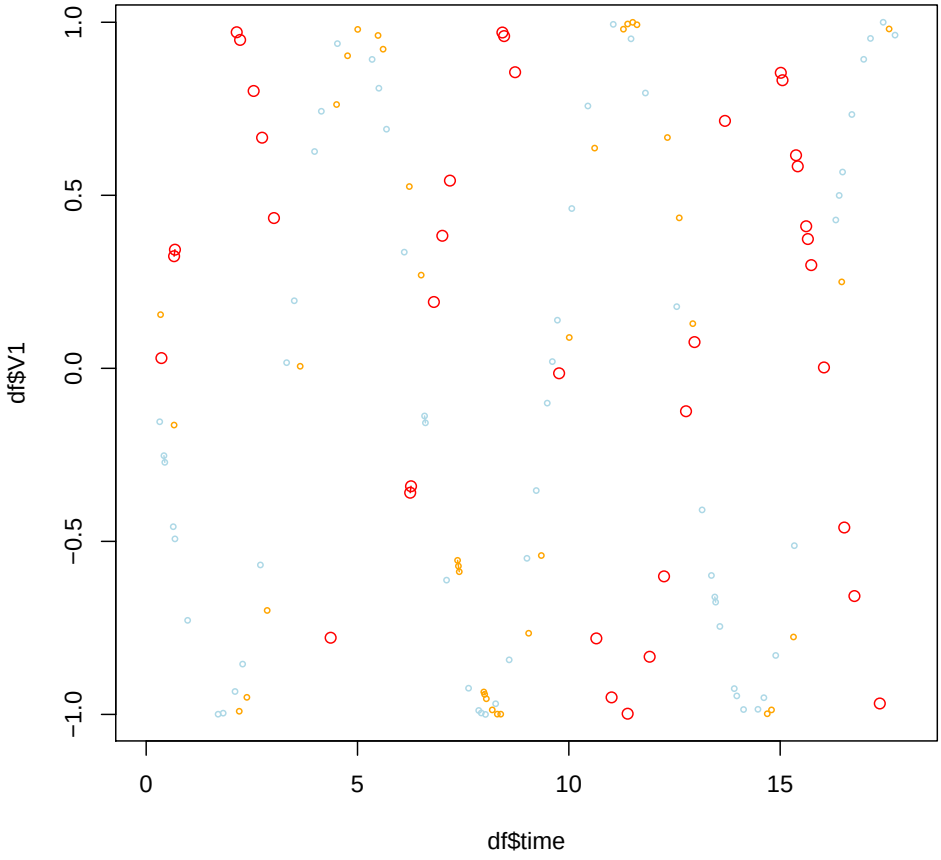
```
## V28                NA          NA       NA        NA
## V29                NA          NA       NA        NA
## V30                NA          NA       NA        NA
## V31                NA          NA       NA        NA
## V32                NA          NA       NA        NA
## V33                NA          NA       NA        NA
## V34                NA          NA       NA        NA
## V35                NA          NA       NA        NA
## V36                NA          NA       NA        NA
## V37                NA          NA       NA        NA
## V38                NA          NA       NA        NA
## V39                NA          NA       NA        NA
## V40                NA          NA       NA        NA
## V41                NA          NA       NA        NA
## V42                NA          NA       NA        NA
## V43                NA          NA       NA        NA
## V44                NA          NA       NA        NA
## V45                NA          NA       NA        NA
## V46                NA          NA       NA        NA
## V47                NA          NA       NA        NA
## V48                NA          NA       NA        NA
## V49                NA          NA       NA        NA
## V50                NA          NA       NA        NA
##
## Residual standard error: NaN on 0 degrees of freedom
## Multiple R-squared:    1,Adjusted R-squared:  NaN
## F-statistic:  NaN on 16 and 0 DF,  p-value: NA

## Results are... horrible.
```

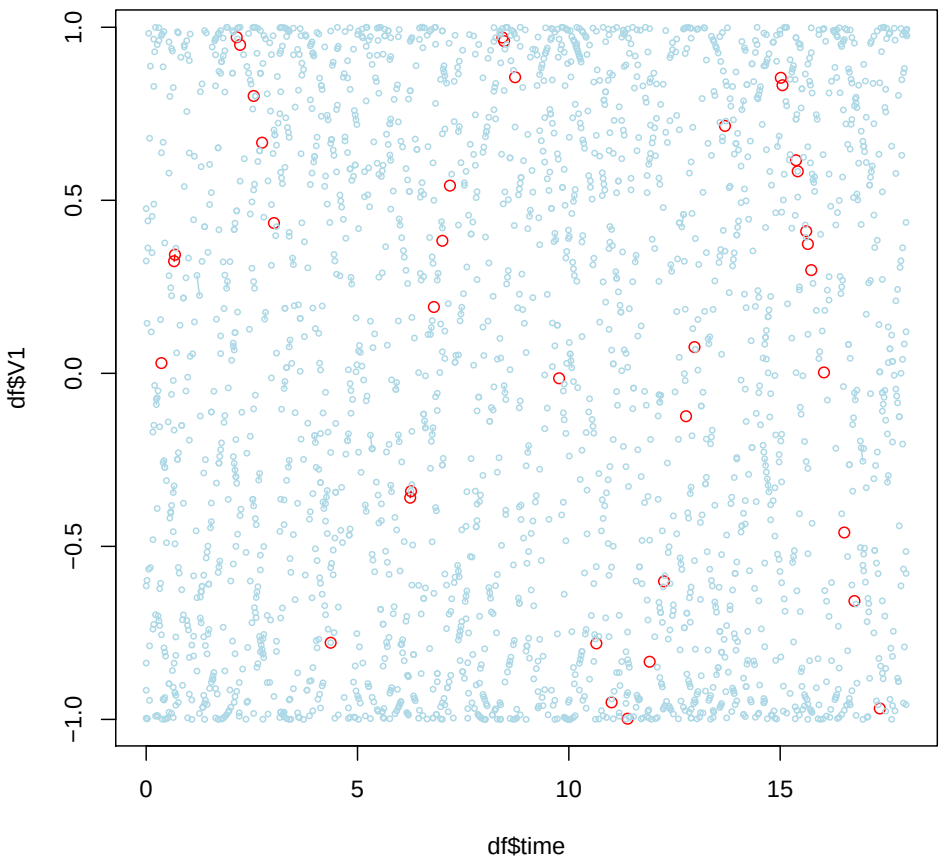
Ok, time to take a different angle. First, I'll re-organize the data, then explore it and see what idea comes up.

```
## Re-order the data in a data.frame
time <- sort(unique(unlist(lapply(trainData, function(x) {
    x[1, ]
}))))
df <- data.frame(time)
df <- cbind(df, matrix(NA, nrow = nrow(df), ncol = length(trainData)))
colnames(df) <- c("time", paste("V", 1:length(trainData), sep = ""))
for (i in 1:length(trainData)) {
    var <- paste("V", i, sep = "")
    df[which(time %in% trainData[[i]][1, ]), var] <- trainData[[i]][2, ]
    ## The previous step assumes that the data is time-sorted
}
```

```
## Exploring the relationship between the variables taking time into account

## First, V1 and V2
plot(df$time, df$V1, type = "o", col = "red")
i <- 2
var <- paste("V", i, sep = "")
lines(df$time, df[, var], type = "o", col = "light blue", cex = 0.5)

## Now: V1, V2 and V3
plot(df$time, df$V1, type = "o", col = "red")
for (i in 2:3) {
    var <- paste("V", i, sep = "")
    lines(df$time, df[, var], type = "o", col = c(NA, "light blue", "orange")[i], cex = 0.5)
}

## All of them
plot(df$time, df$V1, type = "o", col = "red")
for (i in 2:length(trainData)) {
    var <- paste("V", i, sep = "")
    lines(df$time, df[, var], type = "o", col = "light blue", cex = 0.5)
}
```


df$V1
1.0
0.5
0.0
−0.5
−1.0
0
5
10
15
df$time


df$V1
1.0
0.5
0.0
−0.5
−1.0
0
5
10
15
df$time



From the previous plots, now I want to try using something related to *workers*. The idea is to have a function that takes as input the variable number ($iin 2, 3, \ldots, 50$) and the time t. I'll use the times where I have data from variable 1 to create a complete data set. Then, using that set I'll try to predict the values for variable one for $t = 18, 19, 20$. Note that for that I'll use the function to predict the values for the other variables at the same time points before predicting the value of variable 1.

It sounds like lots of prediction is involved and could be a rather fragile method. Hopefully it'll produce better results that my first approach.

```
## Will use splines for the vars 2, 3, ..., 50
library(splines)

## Determine the actual number of splines Note that I don't mind overfitting if it means
## that each model for a specific var will have less bias since I will in a way average
## them at the end

par(mfrow = c(2, 3))
for (n in c(2, 3, 6, 8, 10, 12)) {
    print(paste("Using df equal to", n))
```

```
    fitV2 <- lm(V2 ~ bs(time, df = n), data = df)
    print(summary(fitV2))
    print("******************")
    newV2 <- seq(0, 20, length.out = 300)

    plot(df$time, df$V2, type = "o", main = paste("df equal", n), xlim = c(0, 20))
    lines(newV2, predict(fitV2, data.frame(time = newV2)))
}
```

```
## [1] "Using df equal to 2"
```

```
## Warning:  'df' was too small; have used 3
```

```
##
## Call:
## lm(formula = V2 ~ bs(time, df = n), data = df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -0.8864 -0.5938  0.0085  0.4306  1.4139
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)         -0.896      0.293   -3.06  0.00340 **
## bs(time, df = n)1    2.913      0.903    3.23  0.00210 **
## bs(time, df = n)2   -1.817      0.626   -2.90  0.00526 **
## bs(time, df = n)3    2.031      0.496    4.10  0.00014 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.638 on 56 degrees of freedom
##   (777 observations deleted due to missingness)
## Multiple R-squared: 0.242,Adjusted R-squared: 0.201
## F-statistic: 5.95 on 3 and 56 DF,  p-value: 0.00136
##
## [1] "******************"
```

```
## Warning:  some 'x' values beyond boundary knots may cause ill-conditioned bases
```

```
## [1] "Using df equal to 3"
##
## Call:
## lm(formula = V2 ~ bs(time, df = n), data = df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -0.8864 -0.5938  0.0085  0.4306  1.4139
##
## Coefficients:
```

```
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)         -0.896      0.293   -3.06  0.00340 **
## bs(time, df = n)1    2.913      0.903    3.23  0.00210 **
## bs(time, df = n)2   -1.817      0.626   -2.90  0.00526 **
## bs(time, df = n)3    2.031      0.496    4.10  0.00014 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.638 on 56 degrees of freedom
##   (777 observations deleted due to missingness)
## Multiple R-squared: 0.242,Adjusted R-squared: 0.201
## F-statistic: 5.95 on 3 and 56 DF,  p-value: 0.00136
##
## [1] "********************"
## Warning:  some 'x' values beyond boundary knots may cause ill-conditioned bases
## [1] "Using df equal to 6"
##
## Call:
## lm(formula = V2 ~ bs(time, df = n), data = df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -0.9558 -0.4385  0.0131  0.4141  1.2610
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)        -0.0299     0.4544   -0.07  0.94786
## bs(time, df = n)1  -1.4627     0.9349   -1.56  0.12363
## bs(time, df = n)2   1.4211     0.5556    2.56  0.01342 *
## bs(time, df = n)3  -0.9866     0.6930   -1.42  0.16043
## bs(time, df = n)4   0.5942     0.6382    0.93  0.35610
## bs(time, df = n)5  -1.5651     0.7983   -1.96  0.05518 .
## bs(time, df = n)6   2.6880     0.7295    3.68  0.00054 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.588 on 53 degrees of freedom
##   (777 observations deleted due to missingness)
## Multiple R-squared: 0.389,Adjusted R-squared: 0.32
## F-statistic: 5.63 on 6 and 53 DF,  p-value: 0.00014
##
## [1] "********************"
## Warning:  some 'x' values beyond boundary knots may cause ill-conditioned bases
## [1] "Using df equal to 8"
##
```

```
## Call:
## lm(formula = V2 ~ bs(time, df = n), data = df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -0.3904 -0.2045 -0.0344  0.2110  0.4132
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)         1.1638     0.2936    3.96  0.00023 ***
## bs(time, df = n)1  -3.9497     0.5927   -6.66  1.8e-08 ***
## bs(time, df = n)2   0.0311     0.2998    0.10  0.91793
## bs(time, df = n)3  -0.7504     0.4199   -1.79  0.07989 .
## bs(time, df = n)4  -2.7929     0.3236   -8.63  1.5e-11 ***
## bs(time, df = n)5   1.4698     0.4041    3.64  0.00064 ***
## bs(time, df = n)6  -5.0179     0.3841  -13.07  < 2e-16 ***
## bs(time, df = n)7   1.5045     0.4173    3.61  0.00071 ***
## bs(time, df = n)8  -0.9559     0.4479   -2.13  0.03767 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.264 on 51 degrees of freedom
##   (777 observations deleted due to missingness)
## Multiple R-squared: 0.881,Adjusted R-squared: 0.863
## F-statistic: 47.4 on 8 and 51 DF,  p-value: <2e-16
##
## [1] "********************"

## Warning:  some 'x' values beyond boundary knots may cause ill-conditioned bases

## [1] "Using df equal to 10"
##
## Call:
## lm(formula = V2 ~ bs(time, df = n), data = df)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.11169 -0.03926  0.00285  0.03869  0.11311
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)         0.0743     0.1134    0.66  0.51545
## bs(time, df = n)1  -0.5767     0.2170   -2.66  0.01060 *
## bs(time, df = n)2  -2.0963     0.0944  -22.20  < 2e-16 ***
## bs(time, df = n)3   2.1913     0.1533   14.29  < 2e-16 ***
## bs(time, df = n)4  -0.9002     0.1164   -7.73  4.9e-10 ***
## bs(time, df = n)5  -1.2906     0.1318   -9.79  4.1e-13 ***
## bs(time, df = n)6   2.2636     0.1270   17.83  < 2e-16 ***
```

```
## bs(time, df = n)7   -1.6575     0.1297  -12.78  < 2e-16 ***
## bs(time, df = n)8   -0.5832     0.1469   -3.97  0.00023 ***
## bs(time, df = n)9    1.5239     0.1450   10.51  3.8e-14 ***
## bs(time, df = n)10   0.5907     0.1563    3.78  0.00043 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.0668 on 49 degrees of freedom
##   (777 observations deleted due to missingness)
## Multiple R-squared: 0.993,Adjusted R-squared: 0.991
## F-statistic:  669 on 10 and 49 DF,  p-value: <2e-16
##
## [1] "********************"

## Warning:  some 'x' values beyond boundary knots may cause ill-conditioned bases

## [1] "Using df equal to 12"
##
## Call:
## lm(formula = V2 ~ bs(time, df = n), data = df)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.03173 -0.01270 -0.00058  0.01283  0.03392
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)          0.1309     0.0404    3.24   0.0022 **
## bs(time, df = n)1   -0.5258     0.0694   -7.57  1.1e-09 ***
## bs(time, df = n)2   -1.8585     0.0347  -53.55  < 2e-16 ***
## bs(time, df = n)3    0.3924     0.0488    8.05  2.2e-10 ***
## bs(time, df = n)4    1.3607     0.0421   32.30  < 2e-16 ***
## bs(time, df = n)5   -1.4016     0.0449  -31.23  < 2e-16 ***
## bs(time, df = n)6   -1.0184     0.0431  -23.63  < 2e-16 ***
## bs(time, df = n)7    1.5470     0.0453   34.12  < 2e-16 ***
## bs(time, df = n)8   -0.0135     0.0450   -0.30   0.7664
## bs(time, df = n)9   -1.8282     0.0448  -40.84  < 2e-16 ***
## bs(time, df = n)10   0.4434     0.0503    8.81  1.6e-11 ***
## bs(time, df = n)11   1.0902     0.0514   21.22  < 2e-16 ***
## bs(time, df = n)12   0.6919     0.0565   12.25  3.1e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.0181 on 47 degrees of freedom
##   (777 observations deleted due to missingness)
## Multiple R-squared: 0.999,Adjusted R-squared: 0.999
## F-statistic: 7.61e+03 on 12 and 47 DF,  p-value: <2e-16
##
```

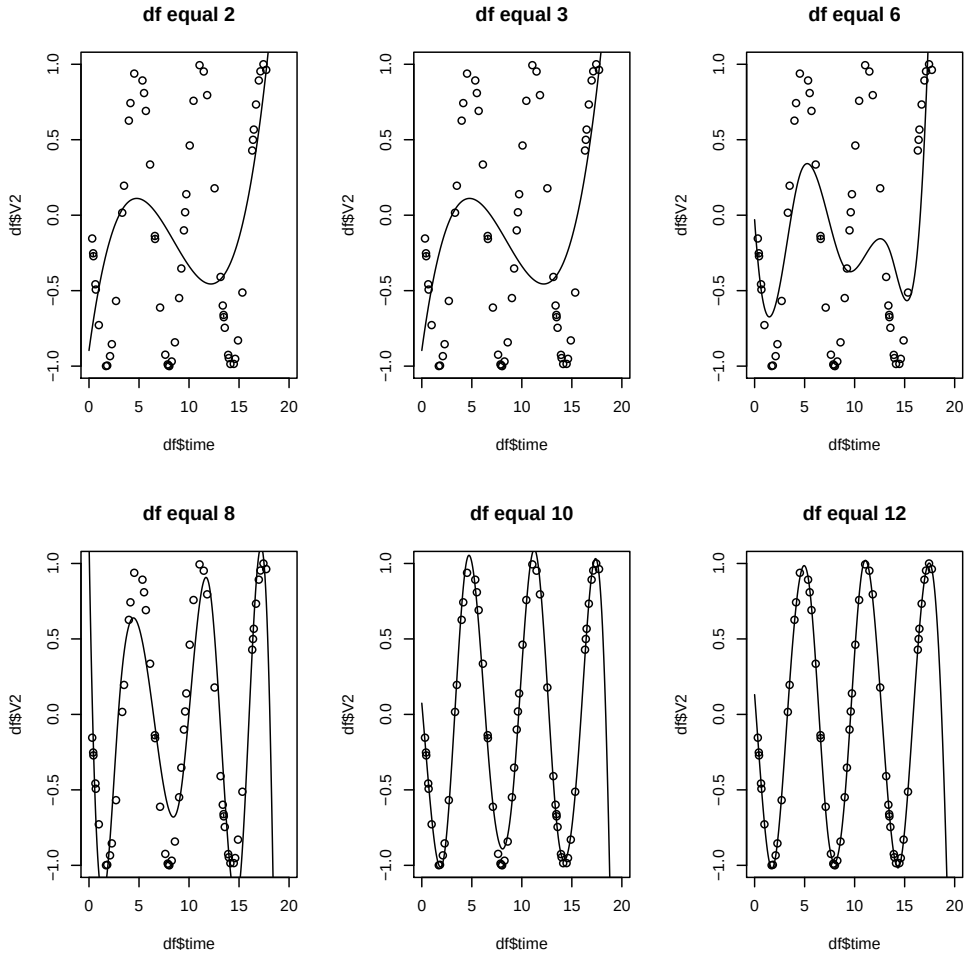


Figure 4: 'Exploring what df parameter to use for the natural splines function ns'

```
## [1] "********************"

## Warning:  some 'x' values beyond boundary knots may cause ill-conditioned bases
```

From figure 4 I decided to use `df` = 10 in the `bs` call. Other exploration releaved that using `ns` instead of `bs` produced poor results.

```
## Create models for vars 2, 3, ..., 50
findT <- df$time[!is.na(df$V1)]
splineFits <- lapply(2:50, function(j) {
    var <- paste("V", j, sep = "")
    lm(df[, var] ~ bs(time, df = 10), data = df)
})
splineT <- lapply(splineFits, function(x) {
    predict(x, data.frame(time = findT))
})
```

```
## Make complete data frame
comp <- data.frame(time = findT, V1 = df$V1[!is.na(df$V1)])
comp <- cbind(comp, matrix(NA, nrow = nrow(comp), ncol = length(splineT)))
colnames(comp) <- c("time", paste("V", 1:length(trainData), sep = ""))
for (i in 1:length(splineT)) {
    comp[, i + 2] <- splineT[[i]]
}

fitIdea2 <- lm(V1 ~ ., data = comp)
summary(fitIdea2)
```

```
##
## Call:
## lm(formula = V1 ~ ., data = comp)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.12799 -0.02209  0.00103  0.03183  0.07742
##
## Coefficients: (40 not defined because of singularities)
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.01925    0.03835    0.50    0.620
## time        -0.00226    0.00978   -0.23    0.819
## V2           0.76956    2.66050    0.29    0.775
## V3          -1.18557    4.58650   -0.26    0.798
## V4           0.09664    1.14359    0.08    0.933
## V5          -0.13817    0.07968   -1.73    0.095 .
## V6           0.53537    7.72724    0.07    0.945
## V7           0.49816    5.88519    0.08    0.933
## V8           0.31188    4.12195    0.08    0.940
## V9          -0.09183    0.59597   -0.15    0.879
## V10         -1.25274    1.66361   -0.75    0.458
## V11               NA         NA      NA       NA
## V12               NA         NA      NA       NA
## V13               NA         NA      NA       NA
## V14               NA         NA      NA       NA
## V15               NA         NA      NA       NA
## V16               NA         NA      NA       NA
## V17               NA         NA      NA       NA
## V18               NA         NA      NA       NA
## V19               NA         NA      NA       NA
## V20               NA         NA      NA       NA
## V21               NA         NA      NA       NA
## V22               NA         NA      NA       NA
## V23               NA         NA      NA       NA
## V24               NA         NA      NA       NA
## V25               NA         NA      NA       NA
## V26               NA         NA      NA       NA
```

```
## V27                NA         NA      NA       NA
## V28                NA         NA      NA       NA
## V29                NA         NA      NA       NA
## V30                NA         NA      NA       NA
## V31                NA         NA      NA       NA
## V32                NA         NA      NA       NA
## V33                NA         NA      NA       NA
## V34                NA         NA      NA       NA
## V35                NA         NA      NA       NA
## V36                NA         NA      NA       NA
## V37                NA         NA      NA       NA
## V38                NA         NA      NA       NA
## V39                NA         NA      NA       NA
## V40                NA         NA      NA       NA
## V41                NA         NA      NA       NA
## V42                NA         NA      NA       NA
## V43                NA         NA      NA       NA
## V44                NA         NA      NA       NA
## V45                NA         NA      NA       NA
## V46                NA         NA      NA       NA
## V47                NA         NA      NA       NA
## V48                NA         NA      NA       NA
## V49                NA         NA      NA       NA
## V50                NA         NA      NA       NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.0514 on 26 degrees of freedom
## Multiple R-squared: 0.995,Adjusted R-squared: 0.994
## F-statistic:  554 on 10 and 26 DF,  p-value: <2e-16


## Seems like only V2 to V10 are usable Further exploration reveals that this is directly
## related to the df paramater

## Re-adjust model
fitIdea2.b <- lm(V1 ~ time + V2 + V3 + V4 + V5 + V6 + V7 + V8 + V9 + V10, data = comp)  #
step(fitIdea2.b)  ## Could try to variable select

## Start:  AIC=-210.6
## V1 ~ time + V2 + V3 + V4 + V5 + V6 + V7 + V8 + V9 + V10
##
##        Df Sum of Sq    RSS  AIC
## - V6    1   0.00001 0.0688 -213
## - V8    1   0.00002 0.0688 -213
## - V4    1   0.00002 0.0688 -213
## - V7    1   0.00002 0.0688 -213
## - V9    1   0.00006 0.0689 -213
```

```
## - time  1   0.00014 0.0689 -213
## - V3    1   0.00018 0.0690 -212
## - V2    1   0.00022 0.0690 -212
## - V10   1   0.00150 0.0703 -212
## <none>              0.0688 -211
## - V5    1   0.00796 0.0767 -209
##
## Step:  AIC=-212.6
## V1 ~ time + V2 + V3 + V4 + V5 + V7 + V8 + V9 + V10
##
##        Df Sum of Sq    RSS  AIC
## - V4    1   0.00001 0.0688 -215
## - V7    1   0.00010 0.0689 -215
## - time  1   0.00057 0.0694 -214
## - V2    1   0.00087 0.0697 -214
## - V8    1   0.00142 0.0702 -214
## - V9    1   0.00270 0.0715 -213
## <none>              0.0688 -213
## - V5    1   0.00841 0.0772 -210
## - V10   1   0.00898 0.0778 -210
## - V3    1   0.01143 0.0802 -209
##
## Step:  AIC=-214.6
## V1 ~ time + V2 + V3 + V5 + V7 + V8 + V9 + V10
##
##        Df Sum of Sq    RSS  AIC
## - V7    1   0.00009 0.0689 -217
## - time  1   0.00115 0.0700 -216
## - V2    1   0.00184 0.0706 -216
## - V9    1   0.00274 0.0716 -215
## - V8    1   0.00285 0.0717 -215
## <none>              0.0688 -215
## - V5    1   0.00845 0.0773 -212
## - V10   1   0.00985 0.0787 -212
## - V3    1   0.01192 0.0807 -211
##
## Step:  AIC=-216.6
## V1 ~ time + V2 + V3 + V5 + V8 + V9 + V10
##
##        Df Sum of Sq    RSS  AIC
## - time  1   0.00129 0.0702 -218
## - V2    1   0.00200 0.0709 -218
## - V9    1   0.00269 0.0716 -217
## <none>              0.0689 -217
## - V5    1   0.00859 0.0775 -214
## - V8    1   0.01174 0.0806 -213
## - V3    1   0.01285 0.0817 -212
```

```
## - V10   1   0.01389 0.0828 -212
##
## Step:  AIC=-217.9
## V1 ~ V2 + V3 + V5 + V8 + V9 + V10
##
##        Df Sum of Sq    RSS  AIC
## - V2    1   0.00159 0.0718 -219
## - V9    1   0.00166 0.0719 -219
## <none>              0.0702 -218
## - V3    1   0.01164 0.0818 -214
## - V5    1   0.01175 0.0819 -214
## - V8    1   0.01213 0.0823 -214
## - V10   1   0.01380 0.0840 -213
##
## Step:  AIC=-219.1
## V1 ~ V3 + V5 + V8 + V9 + V10
##
##        Df Sum of Sq    RSS  AIC
## - V9    1    0.0013 0.0731 -220
## <none>              0.0718 -219
## - V8    1    0.0123 0.0841 -215
## - V5    1    0.0124 0.0842 -215
## - V3    1    0.0249 0.0967 -210
## - V10   1    0.0276 0.0994 -209
##
## Step:  AIC=-220.4
## V1 ~ V3 + V5 + V8 + V10
##
##        Df Sum of Sq    RSS  AIC
## <none>              0.0731 -220
## - V5    1    0.0124 0.0855 -217
## - V8    1    0.0151 0.0882 -215
## - V3    1    0.0236 0.0967 -212
## - V10   1    0.0282 0.1013 -210
##
## Call:
## lm(formula = V1 ~ V3 + V5 + V8 + V10, data = comp)
##
## Coefficients:
## (Intercept)           V3           V5           V8          V10
##     0.00404     -0.49424     -0.11459      0.59496     -0.84379

## Note how it removed the time variable
summary(fitIdea2.b)

##
## Call:
## lm(formula = V1 ~ time + V2 + V3 + V4 + V5 + V6 + V7 + V8 + V9 +
```

```
##     V10, data = comp)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.12799 -0.02209  0.00103  0.03183  0.07742
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.01925    0.03835    0.50    0.620
## time        -0.00226    0.00978   -0.23    0.819
## V2           0.76956    2.66050    0.29    0.775
## V3          -1.18557    4.58650   -0.26    0.798
## V4           0.09664    1.14359    0.08    0.933
## V5          -0.13817    0.07968   -1.73    0.095 .
## V6           0.53537    7.72724    0.07    0.945
## V7           0.49816    5.88519    0.08    0.933
## V8           0.31188    4.12195    0.08    0.940
## V9          -0.09183    0.59597   -0.15    0.879
## V10         -1.25274    1.66361   -0.75    0.458
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.0514 on 26 degrees of freedom
## Multiple R-squared: 0.995,Adjusted R-squared: 0.994
## F-statistic:  554 on 10 and 26 DF,  p-value: <2e-16

## Ok, good to go
```

```
## First, make the predictions for V2, V3, ... for times 18, 19 and 20.
topred <- data.frame(time = c(18, 19, 20))
splinePred <- lapply(splineFits, function(x) {
    predict(x, topred)
})

## Warning:  some 'x' values beyond boundary knots may cause ill-conditioned bases
## Warning:  some 'x' values beyond boundary knots may cause ill-conditioned bases
## Warning:  some 'x' values beyond boundary knots may cause ill-conditioned bases
## Warning:  some 'x' values beyond boundary knots may cause ill-conditioned bases
## Warning:  some 'x' values beyond boundary knots may cause ill-conditioned bases
## Warning:  some 'x' values beyond boundary knots may cause ill-conditioned bases
## Warning:  some 'x' values beyond boundary knots may cause ill-conditioned bases
## Warning:  some 'x' values beyond boundary knots may cause ill-conditioned bases
## Warning:  some 'x' values beyond boundary knots may cause ill-conditioned bases
## Warning:  some 'x' values beyond boundary knots may cause ill-conditioned bases
## Warning:  some 'x' values beyond boundary knots may cause ill-conditioned bases
## Warning:  some 'x' values beyond boundary knots may cause ill-conditioned bases
```

```
## Warning:  some 'x' values beyond boundary knots may cause ill-conditioned bases
## Warning:  some 'x' values beyond boundary knots may cause ill-conditioned bases
## Warning:  some 'x' values beyond boundary knots may cause ill-conditioned bases
## Warning:  some 'x' values beyond boundary knots may cause ill-conditioned bases
## Warning:  some 'x' values beyond boundary knots may cause ill-conditioned bases
## Warning:  some 'x' values beyond boundary knots may cause ill-conditioned bases
## Warning:  some 'x' values beyond boundary knots may cause ill-conditioned bases
## Warning:  some 'x' values beyond boundary knots may cause ill-conditioned bases
## Warning:  some 'x' values beyond boundary knots may cause ill-conditioned bases
## Warning:  some 'x' values beyond boundary knots may cause ill-conditioned bases
## Warning:  some 'x' values beyond boundary knots may cause ill-conditioned bases
## Warning:  some 'x' values beyond boundary knots may cause ill-conditioned bases
## Warning:  some 'x' values beyond boundary knots may cause ill-conditioned bases
## Warning:  some 'x' values beyond boundary knots may cause ill-conditioned bases
## Warning:  some 'x' values beyond boundary knots may cause ill-conditioned bases
## Warning:  some 'x' values beyond boundary knots may cause ill-conditioned bases
## Warning:  some 'x' values beyond boundary knots may cause ill-conditioned bases
## Warning:  some 'x' values beyond boundary knots may cause ill-conditioned bases
## Warning:  some 'x' values beyond boundary knots may cause ill-conditioned bases
## Warning:  some 'x' values beyond boundary knots may cause ill-conditioned bases
## Warning:  some 'x' values beyond boundary knots may cause ill-conditioned bases
## Warning:  some 'x' values beyond boundary knots may cause ill-conditioned bases
## Warning:  some 'x' values beyond boundary knots may cause ill-conditioned bases
## Warning:  some 'x' values beyond boundary knots may cause ill-conditioned bases
## Warning:  some 'x' values beyond boundary knots may cause ill-conditioned bases
## Warning:  some 'x' values beyond boundary knots may cause ill-conditioned bases
## Warning:  some 'x' values beyond boundary knots may cause ill-conditioned bases
## Warning:  some 'x' values beyond boundary knots may cause ill-conditioned bases
## Warning:  some 'x' values beyond boundary knots may cause ill-conditioned bases
## Warning:  some 'x' values beyond boundary knots may cause ill-conditioned bases
## Warning:  some 'x' values beyond boundary knots may cause ill-conditioned bases
## Warning:  some 'x' values beyond boundary knots may cause ill-conditioned bases
## Warning:  some 'x' values beyond boundary knots may cause ill-conditioned bases
## Warning:  some 'x' values beyond boundary knots may cause ill-conditioned bases
## Warning:  some 'x' values beyond boundary knots may cause ill-conditioned bases
## Warning:  some 'x' values beyond boundary knots may cause ill-conditioned bases
## Warning:  some 'x' values beyond boundary knots may cause ill-conditioned bases
```

```
## Re-organize results in a data.frame similar to comp
compPred <- data.frame(time = topred$time)
compPred <- cbind(compPred, matrix(NA, nrow = nrow(compPred), ncol = length(splinePred)))
colnames(compPred) <- c("time", paste("V", 2:length(trainData), sep = ""))
for (i in 1:length(splinePred)) {
    compPred[, i + 1] <- splinePred[[i]]
}
```

```
## Make predictions for V1
(predV1 <- predict(fitIdea2.b, compPred))

##       1       2       3
## -0.5066  2.4248  8.1971


## The splines do not seem to be working for time points 18, 19, 20 because the V2, ...,
## V10 have values outside the -1, 1 range.
summary(comp[, 3:11])

##        V2                V3                V4                V5                V6
##  Min.   :-0.970   Min.   :-1.038   Min.   :-0.9991   Min.   :-1.048   Min.   :-0.974
##  1st Qu.:-0.640   1st Qu.:-0.795   1st Qu.:-0.5133   1st Qu.:-0.840   1st Qu.:-0.635
##  Median :-0.299   Median :-0.234   Median : 0.0985   Median :-0.426   Median :-0.290
##  Mean   :-0.134   Mean   :-0.172   Mean   : 0.1855   Mean   :-0.154   Mean   :-0.132
##  3rd Qu.: 0.106   3rd Qu.: 0.378   3rd Qu.: 1.0201   3rd Qu.: 0.528   3rd Qu.: 0.154
##  Max.   : 1.079   Max.   : 1.047   Max.   : 1.0775   Max.   : 1.046   Max.   : 1.063
##        V7                V8                V9               V10
##  Min.   :-1.036   Min.   :-1.013   Min.   :-1.7121   Min.   :-1.0272
##  1st Qu.:-0.378   1st Qu.:-0.824   1st Qu.:-0.7305   1st Qu.:-0.7729
##  Median : 0.163   Median : 0.553   Median :-0.2749   Median : 0.2179
##  Mean   : 0.177   Mean   : 0.129   Mean   :-0.0825   Mean   : 0.0472
##  3rd Qu.: 0.836   3rd Qu.: 0.825   3rd Qu.: 0.8556   3rd Qu.: 0.6693
##  Max.   : 1.107   Max.   : 1.083   Max.   : 1.0686   Max.   : 1.0012

summary(compPred[, 2:10])

##        V2                V3                V4                V5                V6
##  Min.   :-6.640   Min.   :-8.237   Min.   :-0.794   Min.   :0.0347   Min.   :-9.214
##  1st Qu.:-4.217   1st Qu.:-5.254   1st Qu.:-0.004   1st Qu.:0.2039   1st Qu.:-6.096
##  Median :-1.793   Median :-2.271   Median : 0.787   Median :0.3730   Median :-2.977
##  Mean   :-2.590   Mean   :-3.288   Mean   : 1.562   Mean   :0.3340   Mean   :-3.956
##  3rd Qu.:-0.565   3rd Qu.:-0.814   3rd Qu.: 2.741   3rd Qu.:0.4836   3rd Qu.:-1.326
##  Max.   : 0.662   Max.   : 0.643   Max.   : 4.694   Max.   :0.5941   Max.   : 0.325
##        V7                V8                V9               V10
##  Min.   :-0.831   Min.   :-0.801   Min.   :-23.76   Min.   :-2.846
##  1st Qu.: 0.020   1st Qu.:-0.687   1st Qu.:-17.45   1st Qu.:-2.033
##  Median : 0.872   Median :-0.573   Median :-11.13   Median :-1.220
##  Mean   : 1.592   Mean   :-0.515   Mean   :-12.89   Mean   :-1.347
##  3rd Qu.: 2.803   3rd Qu.:-0.372   3rd Qu.: -7.46   3rd Qu.:-0.597
##  Max.   : 4.734   Max.   :-0.172   Max.   : -3.78   Max.   : 0.026
```

In conclusion, this method mostly failed. It is note realizing that using a step variable selection would remove the `time` variable. Plus, there are plenty of warnings from using `bs` to predict the V2, ..., V50 values noting that there are x values beyond the boundary knots. For this idea to work, I would need to improve the prediction of the values for V2, ..., V50 for values outside the boundaries.

Anyhow, I think that this is more than enough for an in-class exercise!

- R version 2.15.1 (2012-06-22), `x86_64-apple-darwin9.8.0`
- Locale: `en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8`
- Base packages: base, datasets, graphics, grDevices, methods, splines, stats, utils
- Other packages: knitr 0.9
- Loaded via a namespace (and not attached): digest 0.6.0, evaluate 0.4.3, formatR 0.7, stringr 0.6.2, tools 2.15.1