Test1 Spatial Statistics

Drew Remmenga

2022-11-12

- You are encouraged to use web resources and class materials
- Please work alone and hand in your own work.
- Be spare in what you include and you will lose credit if you include too much extraneous output or information. All questions count for an equal number of points.
- Any subproblems marked GRAD are required for 500 level students but will serve as an extra credit question for the 400 level students.
- Please send me email if you have questions or any concerns. `nychka@mines.edu`
- Hand in your work in pdf format in Gradescope. You can keep the questions as part of what you hand in but you should begin your *answer* on a separate page. You can use `\newpage` to create a page break in your work.

Reformatting text and code that are not your answers

It is harder to grade homework when all the introduction and question text are included. But you may also want to keep the question description together with your work. To address both issues you can *comment out* the question portions without just deleting them or put your answers in a different color.

To use the html commenting format:

```
<!--
This text is now commented out and will not be part of the rendered output.
-->
```

To change the color you need to have Latex working:

```
\textcolor{magenta}{This is a magenta block of text. }
```

And the rendered version in pdf:

This is a magenta block of text.

Points

All subsections of the problems count equally for 10 points except for Problem 4(b) which is 20 total.

- 400 level 1(10) 2(10) 3(20) 4(40) 5(20) 6(10) = 110
- 500 level 1(10) 2(20) 3(20) 4(40) 5(20) 6(30) = 140

Some setup

Change the working directory below to where you have downloaded the data and rmd file.

```
suppressMessages(library( fields))
```

```
## Warning: package 'fields' was built under R version 4.1.3
```

```
## Warning: package 'spam' was built under R version 4.1.3
```

```
suppressMessages(library( lubridate))
setwd("~/School/MATH498/Test2")
remove( list=ls())
```

Introduction

The state of Texas has it own power grid that is divided into seven subregions. The data in this test will just use the South Central region that contains Austin and San Antonio. The data is for 2017 and 2018 in separate matrices and the power demand (I think in megawatts, MW) for this region is recorded every 15 minutes. Amazingly there is no missing data in this record.

The goal in this test is to forecast next days power demand based on todays values. This is the same strategy used for the Golden hourly ozone and you should refer to the scripts:

`InClassForecastGoldenOzone.Rmd`, and

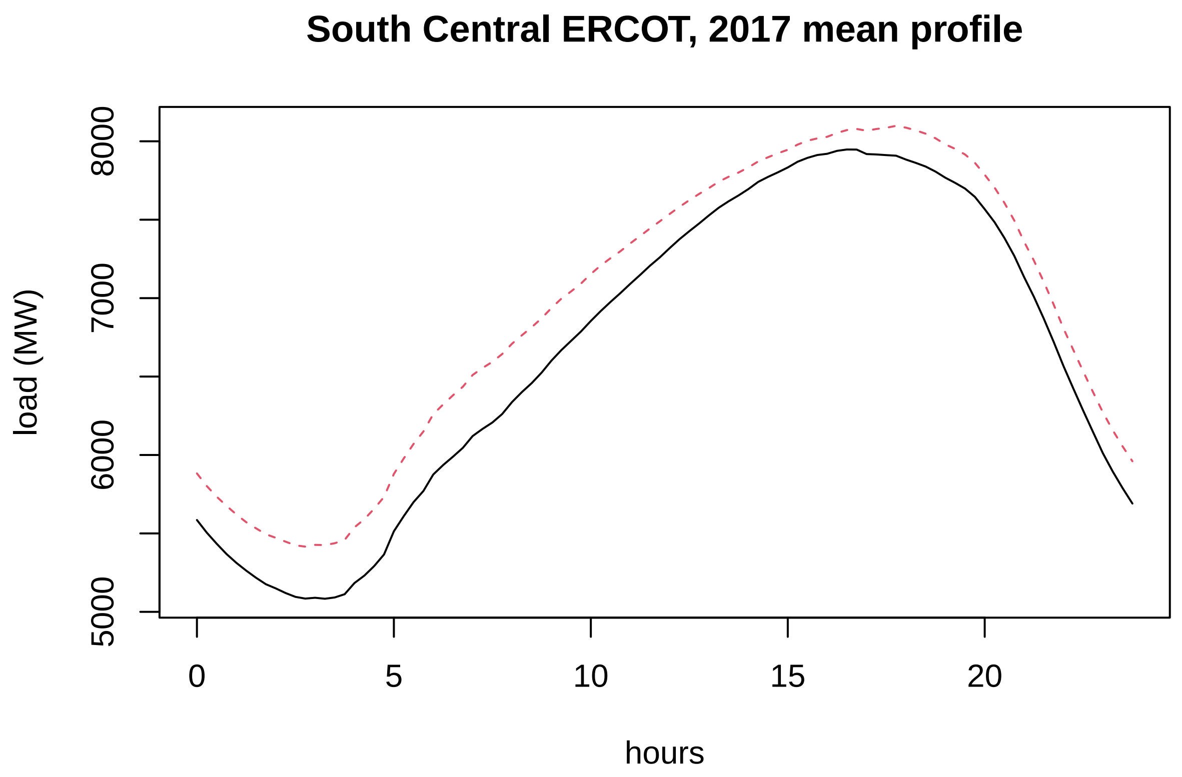
~~~
for code examples.  The script ```ForecastGoldenDaily.R``` is
a handy worked example of the in-class activity. The CV
exercise is not asked for in this test but the code may also
provide more examples of fitting the forecast model.

Datasets are loaded as

```r
load("ERCOTSCentral.rda")
ls()
~~~

```
## [1] "dailyHour" "load2017"  "load2018"
```

- **dailyHour** The 96 15 minute times during the day in hours. I don't think there is an adjustment for daylight savings time.
- **load2017** A 365X96 matrix of the power load. Rows index *days* and columns index the *15 minutes intervals* during each day. Beginning day is Jan 1 and ending day is Dec 31.
- **load2018** Same format but for the year 2018

Here is a plot of the mean power over the day for both years

```
meanProfile1<- colMeans( load2017)
meanProfile2<- colMeans( load2018)
matplot(dailyHour, cbind(meanProfile1,meanProfile2),
                         , type="l", xlab="hours", ylab="load (MW)")
title("South Central ERCOT, 2017 mean profile")
```



Problem 1

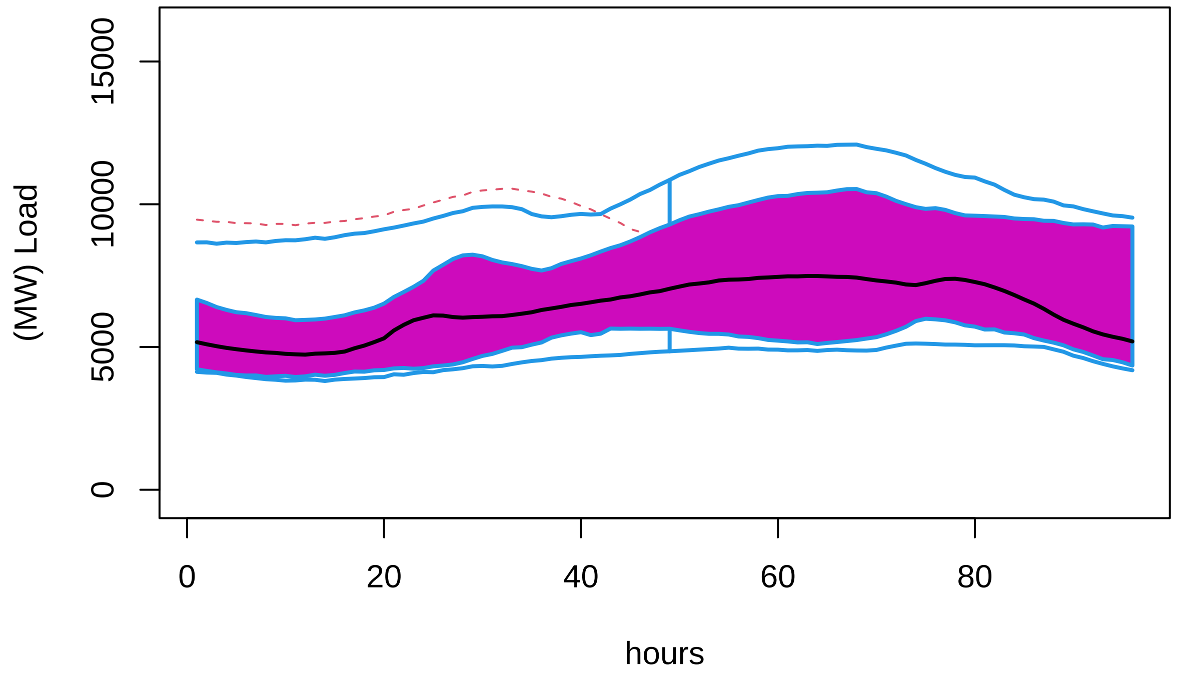
Make a functional bplot over the 96 15 miunte intervals for each day. Add to this the mean Profiles created above. Comment on the shapes of the mean profiles, the deepest curve and the variation about these central curves for the other days.

```r
library(fda)
```

```
## Warning: package 'fda' was built under R version 4.1.3

## Loading required package: splines

## Loading required package: fds

## Warning: package 'fds' was built under R version 4.1.3

## Loading required package: rainbow

## Warning: package 'rainbow' was built under R version 4.1.3

## Loading required package: MASS

## Warning: package 'MASS' was built under R version 4.1.3

## Loading required package: pcaPP

## Warning: package 'pcaPP' was built under R version 4.1.3

## Loading required package: RCurl

## Warning: package 'RCurl' was built under R version 4.1.3

## Loading required package: deSolve

## Warning: package 'deSolve' was built under R version 4.1.3

##
## Attaching package: 'fda'

## The following object is masked from 'package:graphics':
##
##     matplot
```

```r
fbplot(t(load2017), xlab="hours",ylab="(MW) Load", title="FBPlot of 2017 data")
```

```
## Warning in plot.window(...): "title" is not a graphical parameter

## Warning in plot.xy(xy, type, ...): "title" is not a graphical parameter

## Warning in axis(side = side, at = at, labels = labels, ...): "title" is not a
## graphical parameter

## Warning in axis(side = side, at = at, labels = labels, ...): "title" is not a
## graphical parameter
## Warning in box(...): "title" is not a graphical parameter

## Warning in title(...): "title" is not a graphical parameter
```



```
## $depth
##   [1] 0.08419687 0.13930312 0.29352953 0.46155728 0.40198627 0.14922961
##   [7] 0.20233015 0.26954078 0.32997061 0.32433561 0.30289246 0.38076609
##  [13] 0.39488089 0.22137306 0.15872676 0.21292745 0.36607535 0.43629572
##  [19] 0.34110953 0.29491021 0.15168959 0.08249238 0.30369484 0.33969200
##  [25] 0.17396445 0.39751744 0.42287104 0.36508778 0.29417949 0.32286806
##  [31] 0.30322332 0.28065257 0.32477436 0.48172716 0.46104828 0.21544734
##  [37] 0.33514994 0.41645121 0.31632687 0.22464766 0.32408629 0.33743837
##  [43] 0.33464486 0.33101526 0.26093915 0.36325204 0.30468319 0.29167200
##  [49] 0.23306802 0.30814156 0.19024663 0.20421026 0.29326202 0.33267177
##  [55] 0.28925702 0.10718567 0.21447765 0.36890681 0.45265176 0.30206358
##  [61] 0.24794944 0.28191361 0.20202218 0.12912856 0.39954495 0.33581590
##  [67] 0.24222443 0.40576233 0.36815680 0.20587696 0.07913217 0.25960080
##  [73] 0.20904696 0.21257072 0.25823988 0.36679117 0.29931930 0.34573502
##  [79] 0.46592732 0.44703118 0.47681504 0.47234573 0.44928669 0.24663791
##  [85] 0.37037107 0.49665233 0.50016747 0.42651303 0.27680265 0.37168244
##  [91] 0.36879328 0.29521677 0.32126392 0.41565667 0.24344799 0.23911903
##  [97] 0.28318280 0.28263633 0.36906769 0.49999012 0.38159215 0.40098130
## [103] 0.48021414 0.45834760 0.45287097 0.43907340 0.44023675 0.41917385
## [109] 0.48914961 0.48881200 0.48199107 0.21816558 0.07998034 0.33572275
## [115] 0.45086448 0.45785617 0.41649779 0.43850780 0.43464628 0.14710033
## [121] 0.37158052 0.45715038 0.45047701 0.42078410 0.39269218 0.37169765
## [127] 0.39214305 0.49451051 0.49542375 0.49943848 0.49276996 0.49583396
## [133] 0.43059957 0.41333703 0.45556036 0.48682871 0.46401334 0.38918896
## [139] 0.38256655 0.48035840 0.39205210 0.44387011 0.43729333 0.37814178
## [145] 0.42401855 0.38452161 0.35793583 0.43028846 0.46856889 0.46608977
## [151] 0.48983485 0.46866140 0.41249451 0.45558263 0.46917353 0.45703027
## [157] 0.43467262 0.39247234 0.37541256 0.39892823 0.44242749 0.41589392
## [163] 0.33641192 0.27983106 0.25332759 0.22019215 0.20341211 0.23685286
## [169] 0.26971060 0.29754033 0.25342419 0.28129249 0.26928330 0.05532312
## [175] 0.34645084 0.45284102 0.39063911 0.39904599 0.37106039 0.23398926
## [181] 0.14759051 0.25487261 0.28721336 0.21853314 0.25117793 0.17054904
## [187] 0.19758392 0.24811644 0.27953924 0.27141822 0.20484329 0.14817336
## [193] 0.22671327 0.24181657 0.17019152 0.28455094 0.36121215 0.23748526
## [199] 0.20978442 0.13812942 0.05760105 0.05314225 0.12267848 0.23909410
## [205] 0.19238421 0.11913527 0.10298748 0.11698498 0.08867229 0.09067392
## [211] 0.10684492 0.11323276 0.17287464 0.24801248 0.32394328 0.19690260
## [217] 0.20924783 0.22459607 0.35977846 0.36457361 0.29243424 0.18046079
## [223] 0.15598562 0.16434765 0.19541811 0.09280963 0.10525679 0.04665767
## [229] 0.03941383 0.05331975 0.16707468 0.19762673 0.14918853 0.11872867
## [235] 0.19927853 0.29893199 0.36708095 0.46766756 0.46878136 0.49689836
## [241] 0.46320046 0.43508910 0.43436575 0.40702870 0.42058840 0.43311788
## [247] 0.41861013 0.34419251 0.45996522 0.47485166 0.47666011 0.45446413
## [253] 0.42699411 0.48344388 0.46325079 0.44323567 0.36407308 0.35921960
## [259] 0.37752224 0.38884571 0.28298444 0.25422955 0.21690517 0.23312948
## [265] 0.33540945 0.42086940 0.42951478 0.39773101 0.44929515 0.46563927
## [271] 0.47378710 0.48558461 0.46347706 0.42628597 0.44991532 0.43353750
## [277] 0.39669546 0.44654304 0.46129917 0.47139862 0.43683937 0.36102131
## [283] 0.37546995 0.37335510 0.45702368 0.44044279 0.45406756 0.44132059
## [289] 0.28607886 0.25524017 0.32112060 0.44201478 0.50007793 0.48137466
## [295] 0.41526278 0.36990849 0.29130821 0.28819559 0.34999279 0.24544430
## [301] 0.25560756 0.23904800 0.30080693 0.26418348 0.41045963 0.49134364
## [307] 0.48045374 0.49870399 0.46256868 0.48558712 0.47715405 0.36497613
## [313] 0.42462821 0.33959509 0.21827880 0.26493270 0.44038963 0.44808304
## [319] 0.47714167 0.48463812 0.49768961 0.36747359 0.14218695 0.39493530
## [325] 0.36923798 0.31180629 0.26970966 0.21312315 0.17259615 0.17691273
## [331] 0.34174836 0.38696999 0.32842638 0.39118621 0.34274753 0.26491498
## [337] 0.32477984 0.48999213 0.41358228 0.46223108 0.32831787 0.27269290
## [343] 0.27258565 0.35269990 0.37055296 0.41898615 0.38101557 0.44284350
## [349] 0.45088972 0.46303095 0.44989604 0.48140979 0.43174143 0.40068368
## [355] 0.43777661 0.42993439 0.38780154 0.35257320 0.34875135 0.49146627
## [361] 0.31538885 0.31664456 0.42121077 0.39063174 0.39542971
##
## $outpoint
## [1] 7
##
## $medcurve
## [1] 87
```

There are a couple of high outliers and low outliers. The deepest curve follows the mean pretty well with some fluctuations. # Problem 2

2(a)

- Compute the SVD for `load2017` and plot the log10 singular values as a function of their ordering from largest to smallest. For clarity let's refer to the SVD as

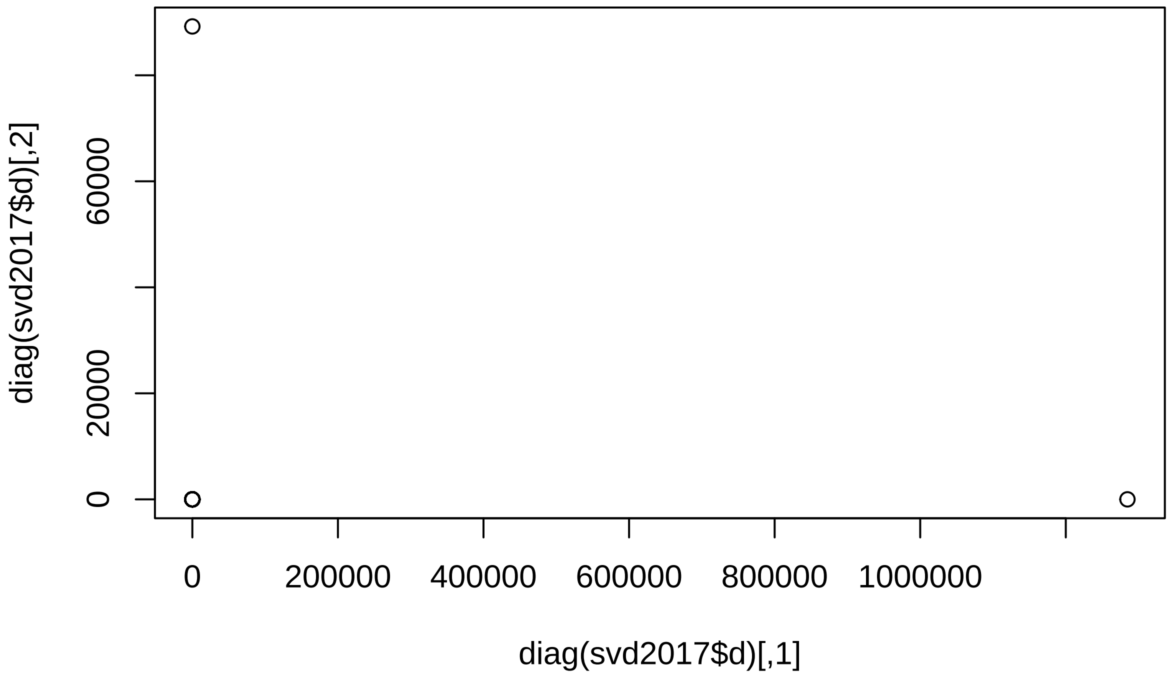
```
load2017 = U %*% D %*%t(V)
```

and here the singular values are `diag(D)`.

- Do you see any obvious breaks in these values where the values become significantly smaller. I.e do you identify a "knee" in your values that is commonly looked for in a singular value plot.
- There are only 96 singular values but 365 days why the mismatch?
- What are the units of the singular values (e.g. mega watts (MW), MW/minute, …)?

The matrices must be square. MW/minute. There is a knee.

```
svd2017=svd(load2017)
plot(diag(svd2017$d))
```



## 2(b) GRAD It is also typical to look at the cumulative effect of the singular values, (D), as `cumsum(D^2 )/ sum(D^2)`. Why square the values here?

Problem 3

Based on the SVD from Problem 2, scale the columns of the V matrix by the singular values to give a "basis function" matrix

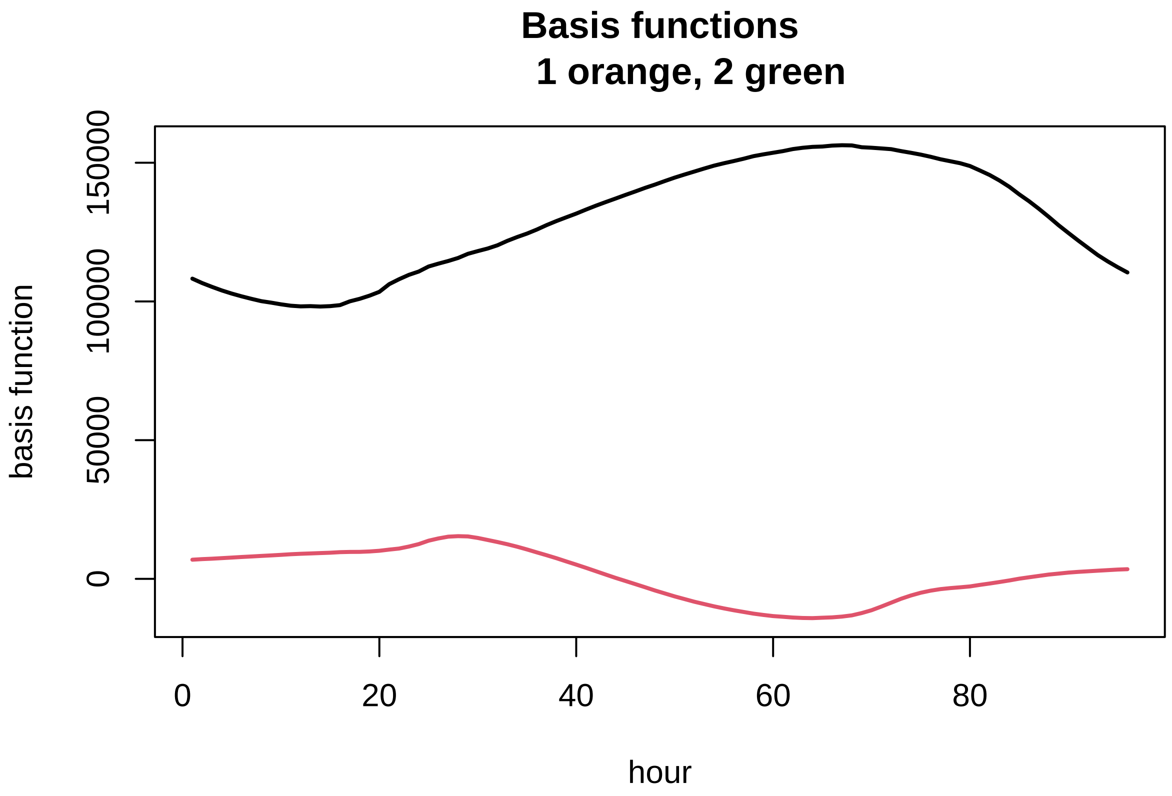
`VB<- V%*%D` where `D` is the diagonal *matrix* of the singular values.

Problem 3(a)

- Plot the first and second basis functions and explain how adding and subtracting the second from the first changes the shape of the daily power demand.
- In answering this question I decided to multiply both U and V by -1. Why did I do this? Will it change any results (e.g. the forecast accuracy) for the rest of this test?

Subtracting the second from the first makes the graph more sinusoidal throughout the day. It will not change the forecast accuracy. This was only to normalize U and V.

```r
V=-1*svd2017$v
D=diag(svd2017$d)
VB=V%*%D
matplot(VB[,1:2], type="l", lty=1, lwd=2,
         xlab="hour", ylab="basis function")
title("Basis functions
      1 orange, 2 green")
```



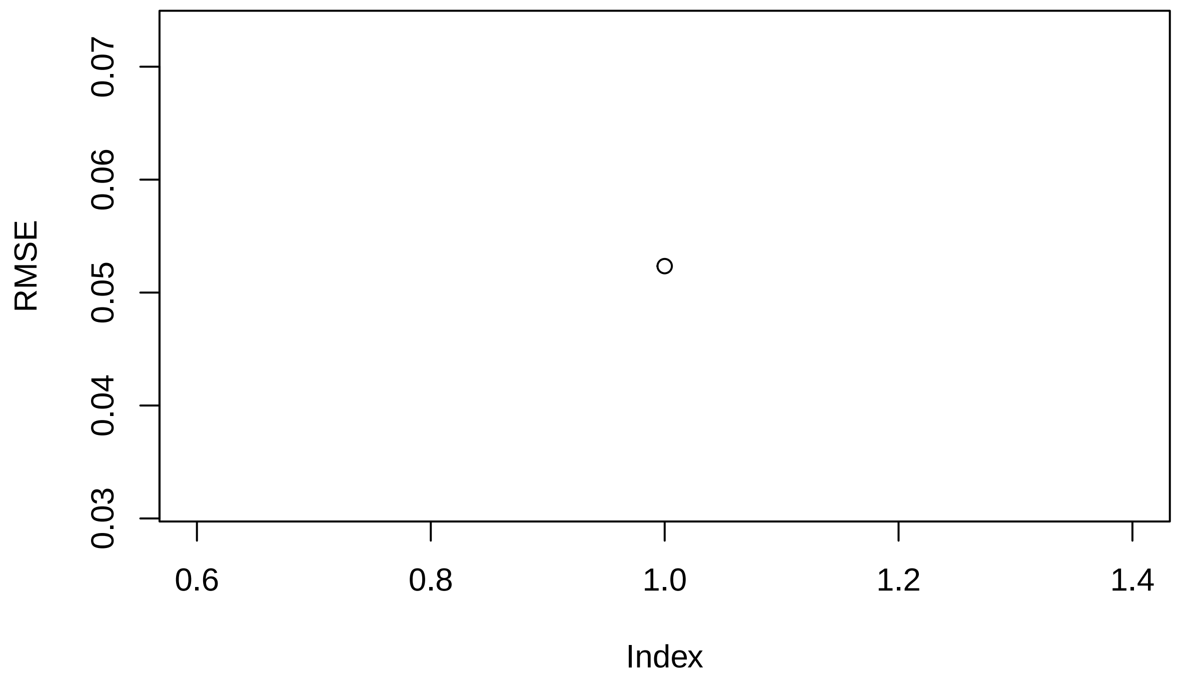
Problem 3(b)

Following the example for forecasting the daily ozone for Golden create an *oracle* forecast for the 96 values of the load demand that assumes you know the first 5 columns of the U matrix for the next day. (This is called on an oracle because we would not know these coefficients just based todays power demand.) We can also find the *average* RMSE by `sqrt(mean (RMSEOracle^2)` for the 365 days. This will be used later in Problem 4.

Find the root mean squared error RMSE of these oracle forecasts by day and plot them over time. Comment on whether the accuracy depends on the time of year.

accuracy depends on time of year.

```
N<- nrow(load2017 )
tm<- dailyHour[-N] # can only forecast up to the second to last day!
Y<- load2017[-1,]
X<-  load2017[-N,]
U=-1*svd2017$u
XU<- U[-N,] # todays coefficients
YU<- U[-1,] # tomorows coefficents
Approxload<- YU[,1:2]%*% t(VB[,1:2])
RMSE=sqrt(mean(U[,1:5]^2))
plot(RMSE)
```



Problem 4

Following the example example for forecasting the daily ozone for Golden create the matrices.

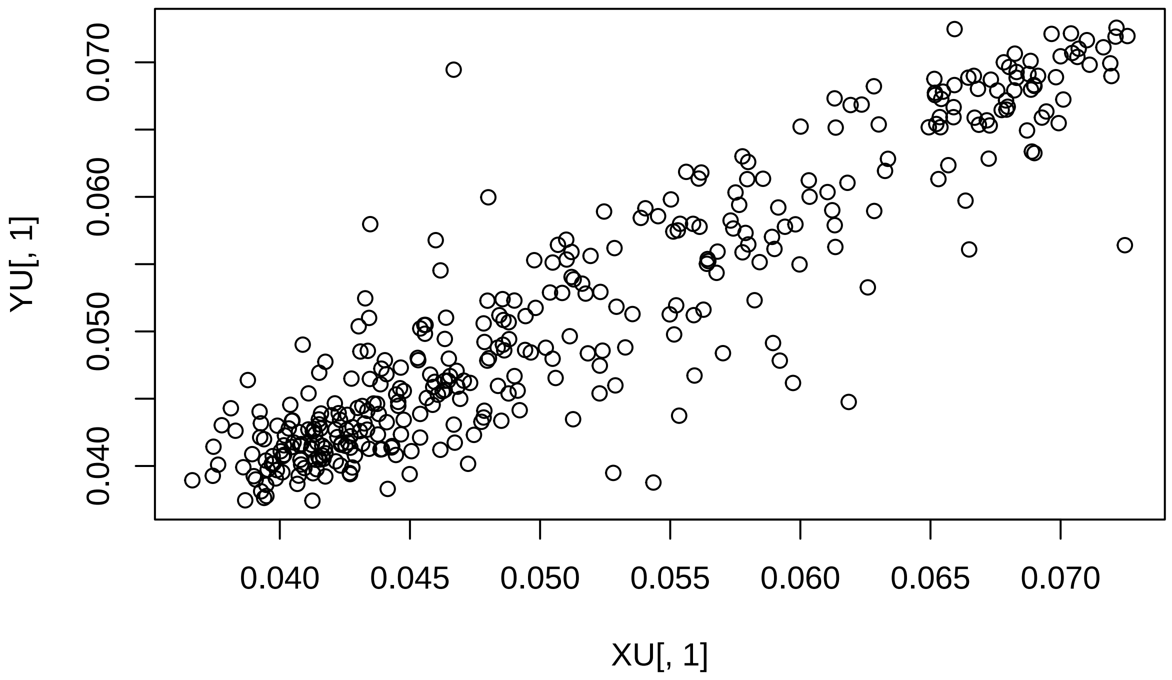
```
N<- 365
YU<- U[-1,]
XU<- U[-N,]
```

4(a)

The plot
`plot( XU[,1] , YU[,1])` suggests some strong dependence. Is this feature *good or bad* for building a model that forecasts tommorrow's power demand based on today's values.

good

```
plot(XU[,1],YU[,1])
```



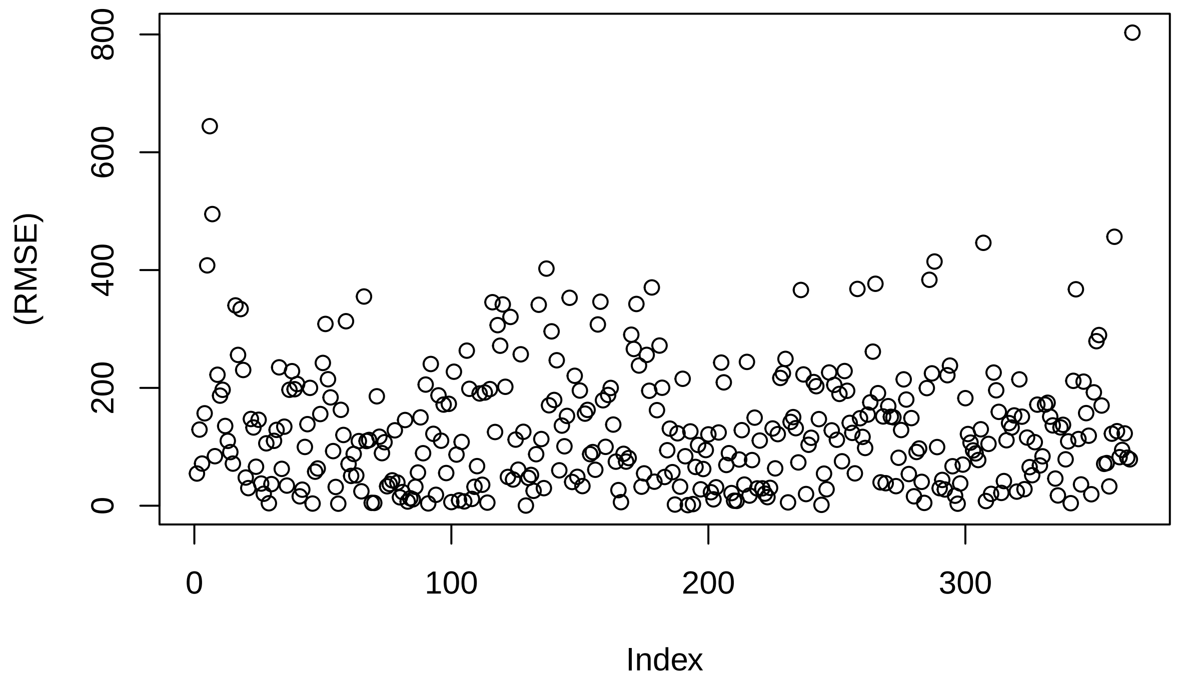
4(b) 20 points

Build a model that forecasts the power demand for the next day based on today's first 5 "U" coefficients and the basis functions.

Plot the RMSE for each day over the year and comment on any patterns in the errors due to seasonality.

Errors are highest in the middle of the year.

```
loadHat<- U[,1:5]%*%t(VB[,1:5])

RMSE<-  sqrt(
              rowSums( load2017 - loadHat)^2
              )
plot((RMSE))
```



## 4(c) Compare the *average* RMSE for this model to the *average* RMSE for the oracle forecasts. Under what circumstances would these forecasts have a smaller RMSE than the oracle ones?

When the svd is a better match for the data. When we use more basis functions.

Problem 5

For your forecasts in Problem 4 find the forecast for the maximum load during the day. If `loadHat5` is your forecast matrix (364X96) then a quick way to find the forecasted maxima and the actual value is

```
maxHat <- apply(loadHat, 1, max )
max2017<- apply(load2017, 1, max )
```

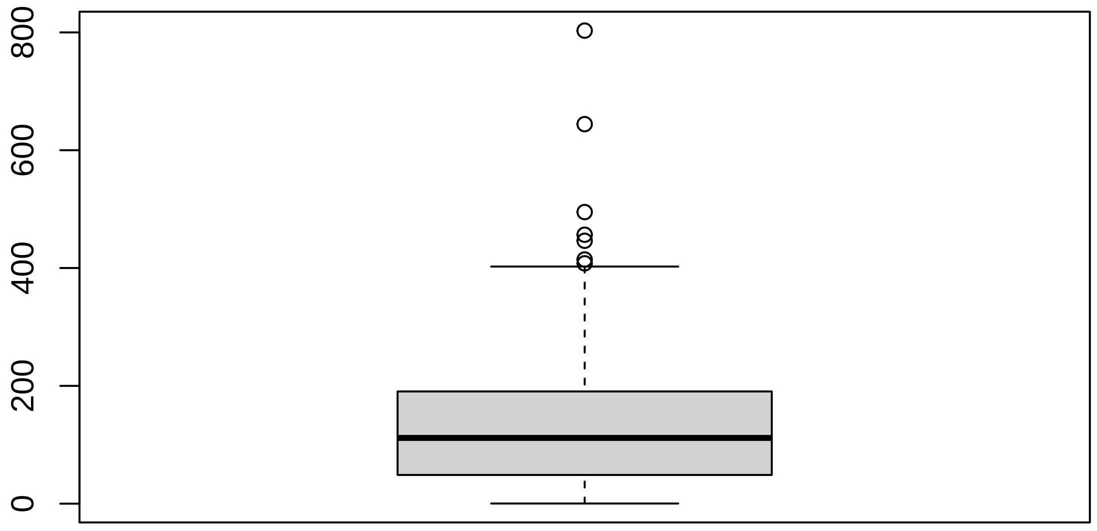
Note that to align these for comparison you need to lag the actual values as `max2017[-1]`. That way the forecast and the actual day are in the same row.

Problem 5(a)

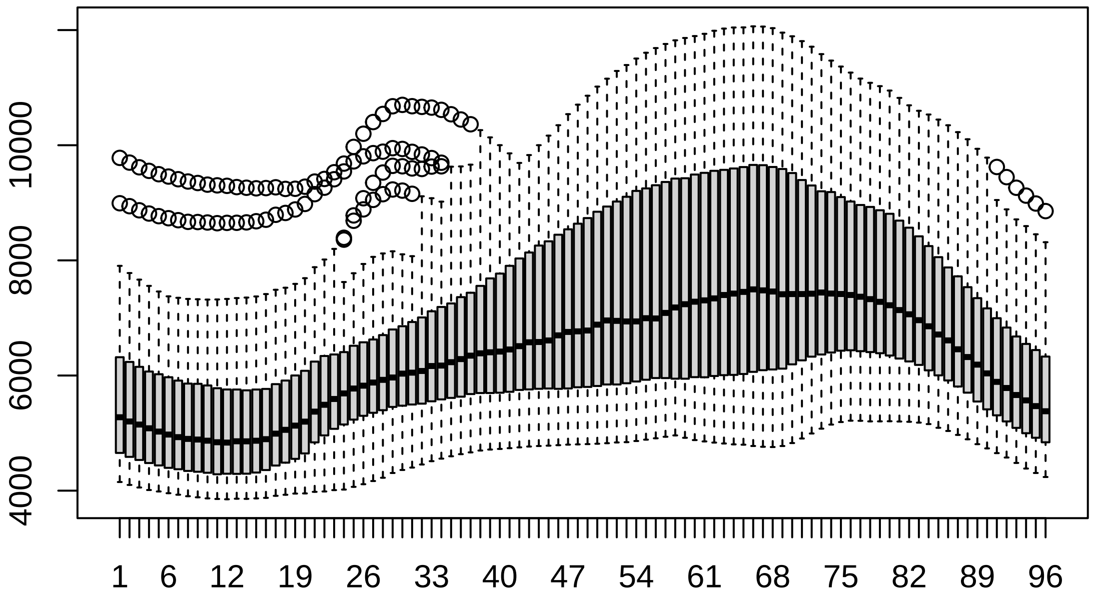
Also find the oracle forecasts. How well do you do in forecasing the daily maximum compared to the functional data approach? Show your results using a figure with two boxplots
of the 364 daily differences between the true value and your forecast from the 5 coefficients functional data model and the differences between the true value and the oracle forecast.

Also report the average RMSE for your forecasts and the oracle.

```
boxplot(RMSE)
```



```
boxplot(loadHat)
```



```r
mean(RMSE)
```

```
## [1] 131.2411
```

```r
mean(loadHat)
```

```
## [1] 6659.741
```

Problem 5(b)

Create a forecast model (using lm) that uses just today's maximum power demand to predict tommorrow's value. Compare your results to the "functional data" forecasts from part (a) by finding the average RMSEs of both.

```
mean(RMSE)
```

```
## [1] 131.2411
```

```
UY<- U[-1,]
UX<- U[-N,]
UX5<- UX[,1:5]
look1<- lm(UY[,1:5] ~  UX5 )
UHat<- look1$fitted.values
O3PC4Hat<- UHat%*%t(VB[,1:5])

RMSEPredict<-  sqrt(
  rowMeans( load2017[-1,] - O3PC4Hat)^2
)
mean(RMSEPredict)
```

```
## [1] 309.5055
```

The second one is better.

Problem 6

Problem 6(a)

Briefly explain the problem of using the RMSEs that you have found in Problems 3-5 as the measures of the forecast accuracy.

RMSE is dependent on seasonal fluctuations. ## Problem 6(b) GRAD
The data set `load2018` are the power demands in the same format for 2017. Apply your forecast model from Problem 4 to see how well it does on these new data. Do not refit your model or basis functions – i.e. use the coefficients fit from lm for the 2017 data. To find the "U" coefficients for these new data you can use

```
U2018<- load2018%*% VB
```

Report the average RMSE for this year.

Problem 6(c) GRAD

What is the advantage of using the SVD from 2017 to do find the forecast accuracy for 2018?