Pipeline of FPCA: Light Curve Fitting and Dimension Reduction

From Dept. of Statistics MediaWiki

(Back to Page of Xiaomeng Yan)

This page is a tutorial which introduce how to use FPCA to do the light curve fitting and dimension reduction on Photometric Data Release 2 (Strizinger et al., 2011)

Contents

1. Data Cleaning

The raw dataset we download is named CSP_Photometry_DR2.tar.gz (http://csp.obs.carnegiescience.edu/data/CSP_Photometry_DR2.tar.gz%7C). The dataset releases 85 nearby Type Ia supernovae's photometry in the form of ".dat".

Each dataset consists of basic information of the observation including the name of SNe Ia, Zcmb, RA and DEC and photometry measurements under the filters u,B, V, g, r, i and measurement's uncertainty. The following is an example of the raw data file,

```
# CSP Optical + NIR data for SN2004dt
# Timestamp:  2011-10-25 15:53:07
# zcmb = 0.018812          RA = 02:02:12.77       DEC = -00:05:51.50
#  All magnitudes in the CSP natural system.  99.9 means 'blank'
#       MJD            u     +/-        B    +/-        V    +/-        g    +/-        r    +/-        i    +/-
53249.26997      16.630  0.013   15.769  0.006   15.576  0.006   15.593  0.004   15.660  0.004   16.069  0.006
53250.29201      16.782  0.012   15.848  0.008   15.629  0.009   15.661  0.008   15.701  0.007   16.110  0.009
53251.31405      16.886  0.011   15.949  0.009   15.672  0.009   15.768  0.008   15.753  0.009   16.110  0.009
53252.34789      17.030  0.011   16.047  0.006   15.719  0.005   15.835  0.005   15.786  0.004   16.049  0.007
53253.26951      17.150  0.011   16.151  0.013   15.774  0.005   15.910  0.004   15.778  0.004   99.900  99.900
53256.22375      17.627  0.009   16.568  0.006   15.960  0.010   16.237  0.005   15.803  0.005   16.021  0.007
53257.21014      17.767  0.013   16.705  0.008   16.016  0.009   16.355  0.008   15.837  0.010   16.032  0.011
53258.24234      17.913  0.011   16.854  0.007   16.080  0.008   16.487  0.005   15.858  0.005   16.008  0.007
53259.25928      18.062  0.011   16.998  0.006   16.159  0.005   16.622  0.004   15.901  0.004   15.993  0.005
53260.28264      18.216  0.011   17.132  0.008   16.239  0.008   16.732  0.008   15.931  0.009   15.992  0.009
53261.19531      18.350  0.013   17.226  0.013   16.315  0.011   16.847  0.008   15.953  0.009   15.982  0.009
53262.19022      18.446  0.014   17.383  0.009   16.402  0.010   16.966  0.011   16.045  0.009   16.018  0.009
53263.24848      18.602  0.016   17.513  0.014   16.524  0.013   17.077  0.012   16.104  0.011   16.082  0.014
53264.22499      18.707  0.019   17.607  0.015   16.590  0.013   17.206  0.010   16.154  0.011   16.124  0.014
53265.26678      18.802  0.015   17.707  0.014   16.664  0.009   17.315  0.008   16.243  0.007   16.158  0.008
53266.21198      18.847  0.015   17.797  0.009   16.742  0.007   17.378  0.005   16.307  0.004   16.231  0.005
53267.22001      18.945  0.022   17.859  0.012   16.805  0.008   17.452  0.008   16.385  0.006   16.289  0.009
53268.22471      19.016  0.018   17.901  0.009   16.851  0.007   17.497  0.009   16.397  0.008   16.338  0.010
53269.17606      99.900  99.900  17.955  0.017   16.928  0.007   99.900  99.900  99.900  99.900  99.900  99.900
53270.18012      99.900  99.900  17.997  0.011   16.955  0.006   99.900  99.900  99.900  99.900  99.900  99.900
53271.27227      19.130  0.023   18.035  0.011   16.997  0.007   17.642  0.008   16.546  0.007   16.532  0.009
53272.25135      19.120  0.026   18.071  0.012   17.030  0.008   17.682  0.009   16.592  0.009   16.565  0.008
53273.27770      19.105  0.032   18.110  0.012   17.062  0.009   17.693  0.008   16.633  0.008   16.623  0.009
```

Step1: Format Data

Our FPCA method is focus on four filters "B, V, R,I", so we take the first step to extract photometric data under these specific filters and format these data such that it can be accessed by SnooPy package.

Object of Snoopy

```
{name} {redshift} {ra: right-ascension} {dec:declination}
filter {filter1}
{Data1} {magnitude1} {error1}
...
{DataN} {magnitudeN} {errorN}
filter {filter2}
...
{DataM} {magnitudeM} {errorM}
```

The filter names have to be recognized by SNooPy (fset.list_filters() to see)

TransferFormat.R

Main Function

```
setwd("~/Dropbox/project/snoopy/pipeline")
rawpath <- "/Users/yanxiaomeng/Dropbox/project/snoopy/pipeline/Data/CSP_Photometry_DR2/"

rawfile <- list.files(rawpath)
options(digits=10)

for(i in 1:length(rawfile)){
  # The path for the specific object
  inputname <- paste0(rawpath,rawfile[i])
  # Read the raw data (without removing "#")
  inputfile_raw <- readLines(inputname)
  # Remove "#"
```

```
  inputfile_all <- gsub("#","",inputfile_raw)
  # Calculate the length of observations
  Nlines = length(inputfile_all)-5
  dataframName  <- unlist(strsplit(gsub(" ","",inputfile_all[5]),split="\t"))
  lightcurveMat <- data.frame(matrix(0,nrow = Nlines,ncol = length(dataframName)))
  for(j in 1:Nlines){
  lightcurveMat[j,] <- as.numeric(unlist(strsplit(inputfile_all[j+5],split="\t")))
  }
  names(lightcurveMat) <- dataframName
  # Then extract the light curve information
  Objname <- unlist(strsplit(inputfile_all[1], split = " "))[8]
  RRADEC <- unlist(strsplit(gsub("\t","",inputfile_all[3]), split = " "))
  Redshift <- as.numeric(RRADEC[4])
  RA <- TranFunRA(RRADEC[8])
  DEC <- TranFunDEC(RRADEC[11])
  FilterName <- c('B','V','r','i')
  # Extract BVRI photometric data
  lightcurveMat <- ExtractBVRIFun(lightcurveMat,FilterName)
  # Format data and write
  WriteFun(Objname,Redshift,RA,DEC,lightcurveMat)
}
```

Function to write data in the form of SnooPy objection

```
# WriteTxt function: combine the data frame and the information about the Object
# which generate the format that can be recognized by SNOOPY
WriteFun <- function(Objname_,Redshift_,RA_,DEC_,lightcurveMat_){
  InfVec <- c(Objname_,Redshift_,RA_,DEC_)
  outpath <- "/Users/yanxiaomeng/Dropbox/project/snoopy/pipeline/Data/FormatedBVRI/"
  Objnametxt <- paste0(outpath,Objname_,"_CSP_main.txt")
  write.table(t(InfVec),Objnametxt,sep = " ",col.names = FALSE,row.names = FALSE,quote = FALSE)
  Nfilter <- (ncol(lightcurveMat_)-1)/2
  for(j in 1:Nfilter){
  newdf <- data.frame(MJD = lightcurveMat_[,1],filter = lightcurveMat_[,2*j],Sig = lightcurveMat_[,2*j+1])
  bol <- unique(c(which(newdf[,2]==99.9),which(newdf[,3]==99.9)))
  if(length(bol)==0){
    reNadf <- newdf
  }else{
  reNadf <- newdf[-unique(c(which(newdf[,2]==99.9),which(newdf[,3]==99.9))),]}
  fileName = names(lightcurveMat_)[j*2]
  write.table(t(c("filter",fileName)),Objnametxt,sep = " ",append = TRUE,col.names = FALSE,row.names = FALSE,
  write.table(reNadf,Objnametxt,sep = " ",append = TRUE,col.names = FALSE,row.names = FALSE,quote = FALSE)
  }
}
```

Function to extract photometric data under specific filters

```
### Extract specific filter information from the lightcurveMat dataframe.
ExtractBVRIFun <- function(lightcurveMat_,FilterName_){
  lightcurveMat_name <- names(lightcurveMat_)
  resdf <- data.frame(MJD = lightcurveMat_[,1])
  for(i in FilterName){
    locFilter <- which(lightcurveMat_name == i)
    if(length(locFilter)>0){
      resdf[,i] <- lightcurveMat_[,locFilter]
      resdf[,ncol(resdf)+1] <- lightcurveMat_[,locFilter+1]
      names(resdf)[ncol(resdf)] <- paste0("sigma",i)
    }else{
      next
    }
  }
  return(resdf)
```

```
}
```

Function to transfer ra and dec

```
# Transfer function which transfer ::: to .
TranFunRA <- function(InputChara){
  tmp <- as.numeric(unlist(strsplit(InputChara,split = "\\:")))
   res <- (tmp[1]+tmp[2]/60+tmp[3]/3600)*15
  return(res)
}

TranFunDEC <-function(InputChara){
  tmp <- as.numeric(unlist(strsplit(InputChara,split = "\\:")))
  res <- abs(tmp[1])+tmp[2]/60+tmp[3]/3600
  if(tmp[1]>=0){
  return(res)
  }else{return((-1)*res)}
}
```

The formatted data is stored in FormatedBVRI file and the following is an example of formatted data

```
SN2004dt 0.018812 30.5532083333333 0.0976388888888889
filter B
53249.26997 15.769 0.006
53250.29201 15.848 0.008
53251.31405 15.949 0.009
53252.34789 16.047 0.006
53253.26951 16.151 0.013
53256.22375 16.568 0.006
53257.21014 16.705 0.008
53258.24234 16.854 0.007
53259.25928 16.998 0.006
53260.28264 17.132 0.008
53261.19531 17.226 0.013
53262.19022 17.383 0.009
53263.24848 17.513 0.014
53264.22499 17.607 0.015
53265.26678 17.707 0.014
53266.21198 17.797 0.009
53267.22001 17.859 0.012
53268.22471 17.901 0.009
53269.17606 17.955 0.017
```

Step2: Select Data

The selection criterion is based on the light curve time and color coverage. Each SN in our sample must have at least one observation within 5 days before the light curve maximum, and at least one observation within 5 days after the maximum and this is required for all four filter bands.

TmaxCalculate.py: Calculate the Tmax for four bands

Before applying selection criterion, we use **TmaxCalculate.py** to calculate Tax for four bands and store Tmax information in the **Tmax.csv**. **TmaxCalculate.py** is based on the function **get_max(filter)** of the class **snpy.get_sn()**. Before doing this, we remove **SN2009dc** because redshift is 0 and **get_max** doesn't work in this case.

```
import snpy
from os import listdir
from os.path import isfile,join
import numpy as np

mypath = "/Users/yanxiaomeng/Dropbox/project/snoopy/pipeline/Data/FormatedBVRI/"
outpath = "/Users/yanxiaomeng/Dropbox/project/snoopy/pipeline/Data/DataSelection/Tmax.csv"

onlyfiles = [f for f in listdir(mypath) if isfile(join(mypath,f))]

FilterName = ["B","V","r","i"]
headname = "SN,B,V,r,i,\n"

fout = open(outpath,'w')
fout.write(headname)

for f in range(1,len(onlyfiles)):
    print(f)
    filepath = mypath + onlyfiles[f]
    s = snpy.get_sn(filepath)
    s.fit()
    currLine = onlyfiles[f] + ","
    for I in range(0,len(FilterName)):
        currLine  = currLine + str(s.get_max(FilterName[I])[0]) + ","
    currLine  = currLine + "\n"
    fout.write(currLine)
fout.close()
```

The following is the first several rows of **Tmax.csv**

	A	B	C	D	E
1	SN	B	V	r	i
2	SN2004dt_CSP_main.txt	53236.4	53237.6	53237.7	53233.9
3	SN2004ef_CSP_main.txt	53264	53265.2	53265.2	53261.6
4	SN2004eo_CSP_main.txt	53278.2	53279.4	53279.4	53275.7
5	SN2004ey_CSP_main.txt	53303.9	53305.1	53305.8	53301.1
6	SN2004gc_CSP_main.txt	53325.6	53326.9	53326.8	53323.2
7	SN2004gs_CSP_main.txt	53354.1	53355.3	53355.1	53351.6
8	SN2004gu_CSP_main.txt	53362.4	53363.2	53364.9	53359.1
9	SN2005A_CSP_main.txt	53379	53380.3	53380.6	53376.3
10	SN2005ag_CSP_main.txt	53413.3	53414.1	53415.7	53410
11	SN2005al_CSP_main.txt	53429.1	53430.4	53430.3	53426.7
12	SN2005am_CSP_main.txt	53436.7	53437.7	53437.5	53434.1
13	SN2005be_CSP_main.txt	53460.8	53461.9	53461.6	53458.3
14	SN2005bg_CSP_main.txt	53469.2	53470.4	53470.9	53466.4

DataSelection.R: apply data selection criterion

After running **TmaxCalculate.py**, we have calculated the maximum date for all four bands and store them in **Tmax.csv**. We use **DataSelection.R** to carry out our selection criterior.

```
## Read all the data
setwd("~/Dropbox/project/snoopy/pipeline/Code/")
BVRIpath <- "/Users/yanxiaomeng/Dropbox/project/snoopy/pipeline/Data/FormatedBVRI/"
BVRIfile <- list.files(BVRIpath)
outputPath <- "/Users/yanxiaomeng/Dropbox/project/snoopy/pipeline/Data/DataSelection/"
options(digits=10)
Tmaxinfor <- read.csv("/Users/yanxiaomeng/Dropbox/project/snoopy/pipeline/Data/DataSelection/Tmax.csv")
FileName <- Tmaxinfor[,1]
```

```
Tmaxlist <- list()
Tmaxlist[[1]] <- Tmaxinfor[,2] # B
Tmaxlist[[2]] <- Tmaxinfor[,3] # V
Tmaxlist[[3]] <- Tmaxinfor[,4] # r
Tmaxlist[[4]] <- Tmaxinfor[,5] # i
## Loop for files
for(i in 1:length(BVRIfile)){
  print(i)
  # The path for the specific object
  # Read the raw data (without removing "#")
  inputfile_BVRI <- readLines(paste0(BVRIpath,BVRIfile[i]))
  # Locate the "filter" position
  LocFilter <- grep("filter",inputfile_BVRI)
  bol <- rep(0,4)
  # Loop for four filters
  for(f in 1:4){
    if(f < 4){
      Nrow <- LocFilter[f+1]-LocFilter[f]-1
    }else{
      Nrow <- length(inputfile_BVRI) - LocFilter[f]
    }
    Fildf <- data.frame(matrix(0,ncol = 3, nrow = Nrow))
    for(j in 1:Nrow){
      Fildf[j,] <- as.numeric(unlist(strsplit(inputfile_BVRI[LocFilter[f]+j],split=" ")))
    }
    MJD_f <- Fildf[,1]
    bol[f] <- DeterSelected(MJD_f, Tmaxlist[[f]][i])
  }
  boolS <- sum(bol)
  if(boolS == 4){
    write.table(inputfile_BVRI,file = paste0(outputPath,BVRIfile[i]),row.names = FALSE,col.names = FALSE,quot
  }else{
    next
  }
}


DeterSelected <- function(MJDSeq_, Tmax_){
  LocP <- which(MJDSeq_<= Tmax_)
  locO <- which(MJDSeq_>=Tmax_)
  if(length(LocP) == 0||length(locO) ==0 ){
    return(0)
  }else{
  PreMJD <- Tmax_-MJDSeq_[max(LocP)]
  OverMJD <- MJDSeq_[min(locO)]-Tmax_
  if(PreMJD<=5&OverMJD<=5){
    return(1)
  }else{
    return(0)
  }
  }
}
```

Finally, we get 34 objects in total that meet our selection criterion.

Step3: K-correction and S-correction

Both K-correction and S-correcton are applied to the data so that all the light curve magnitudes are transformed to the data under rest-frame. These corrections are performed using the SNooPy package of Burns et al. (2010) and the employed SED model is from the work of Hsiao et al. (2007).

```
import snpy
import matplotlib
```

```
# ubertemp.py: module to pick  the CSP or Prieto files
## Here: we pick prieto filters
snpy.ubertemp.template_bands = ['B', 'V', "r", 'i']

#Input variable:
##        s: object obtained by get_sn()
##        ks: s.ks dictionary: a dictionary of computed k-corrections. The index
##                            is the filter name, the calue is an array of k-corrections
##                            one for each observed epoch
##        kspath: store k-corrected data into this path

def kcorr_output(s, ks, kspath):
    ks_keys = ks.keys()
    fout = open(kspath, 'w')
    fout.write("Filter,MJD,Mag,Sigma\n")
    for filterI in range(0, len(ks_keys)):
        ckey = ks_keys[filterI]
        lkey = len(ckey)
        filterName = ckey
        ksdata = ks.get(ks_keys[filterI])
        lc = s.data.get(ks_keys[filterI])
        for lineI in range(0, len(ksdata)):
            currLine = filterName + ","
            currLine = currLine + str(lc.MJD[lineI]) + "," + \
                str(lc.magnitude[lineI] - ksdata[lineI]) + "," + \
                str(lc.e_mag[lineI]) + "\n"
            fout.write(currLine)
    fout.close()

from os import listdir
from os.path import isfile, join

figfolder = "/Users/yanxiaomeng/Dropbox/project/snoopy/pipeline/Figs/004kcorr/" # Figure output path
mypath = "/Users/yanxiaomeng/Dropbox/project/snoopy/pipeline/Data/DataSelection/" # Read in path
ksfolder = "/Users/yanxiaomeng/Dropbox/project/snoopy/pipeline/Data/kCorrected/" # File output path

onlyfiles = [f for f in listdir(mypath) if isfile(join(mypath, f))]


for f in range(1, len(onlyfiles)):
    try:
        print f
        filepath = mypath + onlyfiles[f]
        s = snpy.get_sn(filepath)
        s.replot = 0
        s.fit()
        figpath = figfolder + onlyfiles[f] + "lc.png"
        s.plot()
        matplotlib.pyplot.savefig(figpath)
        figpath = figfolder + onlyfiles[f] + "kcorr.png"
        kspath = ksfolder + onlyfiles[f]
        kspath = kspath.replace("txt", "csv")
        kcorr_output(s, s.ks, kspath)
    except:
        print "ERROR in file: " + str(f)
```

2. Webscraping.py: Web Scrape

Before web scraping, we use a simple code **DataSelection_Zcmb.R** to extract redshift information.

```
# Extract the redshift information from the selected supernova Ia

dataselectionPath <- "/Users/yanxiaomeng/Dropbox/project/snoopy/pipeline/Data/DataSelection/"
dataselectionFile <- list.files(dataselectionPath)
```

```
setwd(dataselectionPath)
SNe <- rep(0,length(dataselectionFile))
Zcmb <- rep(0,length(dataselectionFile))
for(FileI in 1:length(dataselectionFile)){
  infor <- readLines(dataselectionFile[FileI])[1]
  SNe[FileI] <- strsplit(infor,split = " ")[[1]][1]
  Zcmb[FileI] <- as.numeric(strsplit(infor,split = " ")[[1]][2])
}
redshiftdf = data.frame(SNe = SNe,Zcmb = Zcmb)
write.csv(redshiftdf,file = "/Users/yanxiaomeng/Dropbox/project/snoopy/pipeline/Data/Redshift.csv",row.names
```

Considering galactic extinction has influence on photometric data, we use **webscrapying.py** to scrape these extinction data from https://ned.ipac.caltech.edu and generate a supernova table **coresne_table.csv**.

```
from urllib.request import urlopen
import ssl
import re
from os import listdir
import pandas as pd

myPath = "/Users/yanxiaomeng/Dropbox/project/snoopy/pipeline/Data/kCorrected/"
outputfile = "/Users/yanxiaomeng/Dropbox/project/snoopy/pipeline/Data/model_v2/coresne_table.csv"


onlyFiles = [f for f in listdir(myPath) if re.search(r'csv', f)]

nFiles = len(onlyFiles)

colNames = ['SN', 'Survey','snetype', 'Type', 'AB', 'AV',
            'AR', 'AI', 'sdssr', 'sdssi',
            'RAX','DECX','Zcmb']

df = pd.DataFrame(index=range(0, nFiles),
                  columns=colNames,
                  dtype='object')

redshiftdf = pd.read_csv("/Users/yanxiaomeng/Dropbox/project/snoopy/pipeline/Data/Redshift.csv")

for rowI in range(0, nFiles):
    print(onlyFiles[rowI].replace(".csv", ""))
    sneName = onlyFiles[rowI].replace(".csv", "")
    survey = 'CSP'
    snetype = 'main'
    df['SN'][rowI] = sneName
    df['Survey'][rowI] = survey
    df['snetype'][rowI] = snetype
    print(rowI)
    print(sneName)
    # connection
    objName = sneName
    https_path = 'https://ned.ipac.caltech.edu/cgi-bin/objsearch?objname='''
    https_path += objName
    https_path += '&extend=no&hconst=73&omegam=0.27&omegav=0.73&corr_z=1&out_csys=Equatorial&out_equinox=J200
    https_path += 'obj_sort=RA+or+Longitude&of=pre_text&zv_breaker=30000.0&list_limit=5&img_stamp=YES'
    ctx = ssl.create_default_context()
    ctx.check_hostname = False
    ctx.verify_mode = ssl.CERT_NONE
    html = urlopen(https_path,context=ctx)
    html_text = html.read().decode('utf-8')
    #SNE Type
    tmp = re.search(r"SuperNova Type\s+?:\s*?(\w+)",html_text)
    if tmp:
      df['Type'][rowI] = tmp.group(1)
    else:
      df['Type'][rowI] = "-1"
```

```
    # extinction
    tmp = re.search(r"<td>Landolt</td><td>B</td><td>.*?</td><td>\s*?(\d*?\.\d*?)\s*?</td>",
                    html_text)
    if tmp:
        df['AB'][rowI] = tmp.group(1)
    else:
        df['AB'][rowI] = -1
    tmp = re.search(r"<td>Landolt</td><td>V</td><td>.*?</td><td>\s*?(\d*?\.\d*?)\s*?</td>",
                    html_text)
    if tmp:
        df['AV'][rowI] = tmp.group(1)
    else:
        df['AV'][rowI] = -1

    tmp = re.search(r"<td>Landolt</td><td>R</td><td>.*?</td><td>\s*?(\d*?\.\d*?)\s*?</td>",
                    html_text)
    if tmp:
        df['AR'][rowI] = tmp.group(1)
    else:
        df['AR'][rowI] = -1

    tmp = re.search(r"<td>Landolt</td><td>I</td><td>.*?</td><td>\s*?(\d*?\.\d*?)\s*?</td>",
                        html_text)
    if tmp:
        df['AI'][rowI] = tmp.group(1)
    else:
        df['AI'][rowI] = -1

    tmp = re.search(r"<td>SDSS</td><td>r</td><td>.*?</td><td>\s*?(\d*?\.\d*?)\s*?</td>",
                        html_text)
    if tmp:
        df['sdssr'][rowI] = tmp.group(1)
    else:
        df['sdssr'][rowI] = -1

    tmp = re.search(r"<td>SDSS</td><td>i</td><td>.*?</td><td>\s*?(\d*?\.\d*?)\s*?</td>",
                        html_text)
    if tmp:
        df['sdssi'][rowI] = tmp.group(1)
    else:
        df['sdssi'][rowI] = -1

    tmp = re.search(r"\d\dh\d\dm\d+?\.\d+?s",
                        html_text)
    if tmp:
        df['RAX'][rowI] = tmp.group(0)
    else:
        df['RAX'][rowI] = -1

    tmp = re.search(r"[+-]\d\dd\d\dm\d+?\.\d+?s",
                        html_text)
    if tmp:
        df['DECX'][rowI] = tmp.group(0)
    else:
        df['DECX'][rowI] = -1
    df['Zcmb'][rowI] =  redshiftdf['Zcmb'][rowI]

df.to_csv(outputfile, index=False)
```

coresne_table.csv

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	SN	Survey	snetype	Type	AB	AV	AR	AI	sdssr	sdssi	RAX	DECX	Zcmb
2	SN2004ef	CSP	main	Ia	0.199	0.151	0.119	0.083	0.125	0.093	22h42m10.0	+19d43m57.	0.029771
3	SN2004eo	CSP	main	Ia	0.392	0.296	0.234	0.163	0.247	0.183	20h32m54.2	+09d45m26.	0.014734
4	SN2004ey	CSP	main	Ia	0.498	0.377	0.298	0.207	0.314	0.233	21h49m07.8	+00d12m39.	0.014627
5	SN2004gu	CSP	main	Ia	0.096	0.073	0.058	0.04	0.061	0.045	12h46m24.7	+12d13m18.	0.046897
6	SN2005A	CSP	main	Ia	0.109	0.083	0.065	0.045	0.069	0.051	02h30m43.2	-03d09m36.4	0.01834
7	SN2005am	CSP	main	Ia	0.194	0.147	0.116	0.081	0.123	0.091	09h16m12.5	-16d05m42.3	0.008967
8	SN2005bl	CSP	main	Ia	0.105	0.079	0.063	0.044	0.066	0.049	12h04m12.2	+20d41m06.	0.025112
9	SN2005eq	CSP	main	Ia	0.268	0.203	0.16	0.111	0.169	0.126	03h08m49.3	-07d13m24.6	0.028351
10	SN2005hc	CSP	main	Ia	0.119	0.09	0.071	0.049	0.075	0.056	01h56m47.9	-00d27m26.8	0.044983
11	SN2005iq	CSP	main	Ia	0.08	0.061	0.048	0.033	0.05	0.038	23h58m32.5	-18d59m15.3	0.032929
12	SN2005kc	CSP	main	Ia	0.479	0.362	0.287	0.199	0.302	0.224	22h34m07.3	+05d18m35.	0.01389
13	SN2005ke	CSP	main	Ia	0.089	0.067	0.053	0.037	0.056	0.042	03h35m04.3	-25d06m35.1	0.004483
14	SN2005ki	CSP	main	Ia	0.116	0.088	0.069	0.048	0.073	0.054	10h40m28.2	+09d27m48.	0.02037
15	SN2005M	CSP	main	Ia	0.113	0.086	0.068	0.047	0.071	0.053	09h37m32.3	+23d25m33.	0.022972
16	SN2005W	CSP	main	Ia	0.259	0.196	0.155	0.108	0.163	0.121	01h50m45.7	+21d30m45.	0.00795
17	SN2006ax	CSP	main	Ia	0.182	0.138	0.109	0.076	0.115	0.085	11h24m03.4	-12d01m00.0	0.017957

3. FPCA

Retrieved from "http://wiki.stat.tamu.edu/index.php?title=Pipeline_of_FPCA:_Light_Curve_Fitting_and_Dimension_Reduction&oldid=28393"

- This page was last modified on 5 January 2018, at 17:20.
- This page has been accessed 112 times.