STAT 8020 R Lab 20

Whitney

November 18, 2020

Contents

Computer Experiments

Design: Latin hypercube

```r
# install.packages("lhs") # Latin Hypercube Sample Package
library(lhs)
# Generate a good n x k LHD
Best = maximinLHS(n = 30, k = 3, dup = 5)
# "dup" is an integer tuning parameter that determines the number of
# candidate points considered. Larger values should inprove results
# but require more computational resources.

# Display the LHD
Best
```

```
##             [,1]        [,2]       [,3]
##  [1,] 0.38915366 0.693036725 0.20715569
##  [2,] 0.66006965 0.396808574 0.73063579
##  [3,] 0.84114493 0.747031793 0.60703590
##  [4,] 0.19462564 0.496400059 0.86120020
##  [5,] 0.30896903 0.452690471 0.49988913
##  [6,] 0.35025802 0.830213753 0.78767635
##  [7,] 0.78228545 0.256412517 0.82107867
##  [8,] 0.68463877 0.329439439 0.30215035
##  [9,] 0.24872595 0.835730868 0.45358300
## [10,] 0.41070739 0.889848622 0.10226565
## [11,] 0.22346887 0.282992760 0.68082803
## [12,] 0.26929144 0.040503975 0.33525135
## [13,] 0.81371763 0.613843933 0.41261060
## [14,] 0.72610963 0.719211030 0.03418110
## [15,] 0.47483890 0.516383700 0.87280107
## [16,] 0.93154567 0.175383376 0.52736326
## [17,] 0.55801140 0.767763704 0.55344932
## [18,] 0.06200952 0.337932975 0.37226730
## [19,] 0.89512044 0.560620080 0.76632119
## [20,] 0.61848400 0.080645510 0.09704705
## [21,] 0.44514403 0.580919945 0.25033711
## [22,] 0.14644907 0.642742924 0.64360293
## [23,] 0.76349516 0.986873273 0.26868303
## [24,] 0.50792386 0.213362546 0.59907387
## [25,] 0.11218618 0.947103121 0.16710458
## [26,] 0.02039435 0.112977907 0.97778745
## [27,] 0.94511679 0.431508082 0.15379446
## [28,] 0.97337323 0.007197774 0.90678596
## [29,] 0.07764597 0.149125555 0.01176941
## [30,] 0.59104173 0.905028270 0.93341753
```

Analysis: Gaussian Process

```
neuron <- read.table("http://deanvossdraguljic.ietsandbox.net/DeanVossDraguljic/R-data/neuron.txt", head
head(neuron, 10)
```

```
##        gNaFsc    gKdrsc fr
## 1  0.38593729 0.2120652 33
## 2  0.04666927 0.4594742  0
## 3  1.00000000 0.4473344 46
## 4  0.95467637 0.3351407 44
## 5  0.53334929 0.7981310 41
## 6  0.59166751 0.6042714 41
## 7  0.18570301 0.3799469 31
## 8  0.49927784 0.2444170 36
## 9  0.74609113 0.3949591 42
## 10 0.07269414 1.0000000  0
```

```
library(mlegp)
GPFit <- mlegp(neuron[, 1:2], neuron[, 3])
```

```
## no reps detected - nugget will not be estimated
##
## ========== FITTING GP # 1 ==============================
## running simplex # 1...
## ...done
## ...simplex #1 complete, loglike = -104.446501 (convergence)
## running simplex # 2...
## ...done
## ...simplex #2 complete, loglike = -104.446501 (convergence)
## running simplex # 3...
## ...done
## ...simplex #3 complete, loglike = -104.446502 (convergence)
## running simplex # 4...
## ...done
## ...simplex #4 complete, loglike = -104.446501 (convergence)
## running simplex # 5...
## ...done
## ...simplex #5 complete, loglike = -104.446501 (convergence)
##
## using L-BFGS method from simplex #1...
##  iteration: 1,loglike = -104.446501
## ...L-BFGS method complete
##
## Maximum likelihood estimates found, log like =  -104.446501
## creating gp object......done
```

```
summary(GPFit)
```

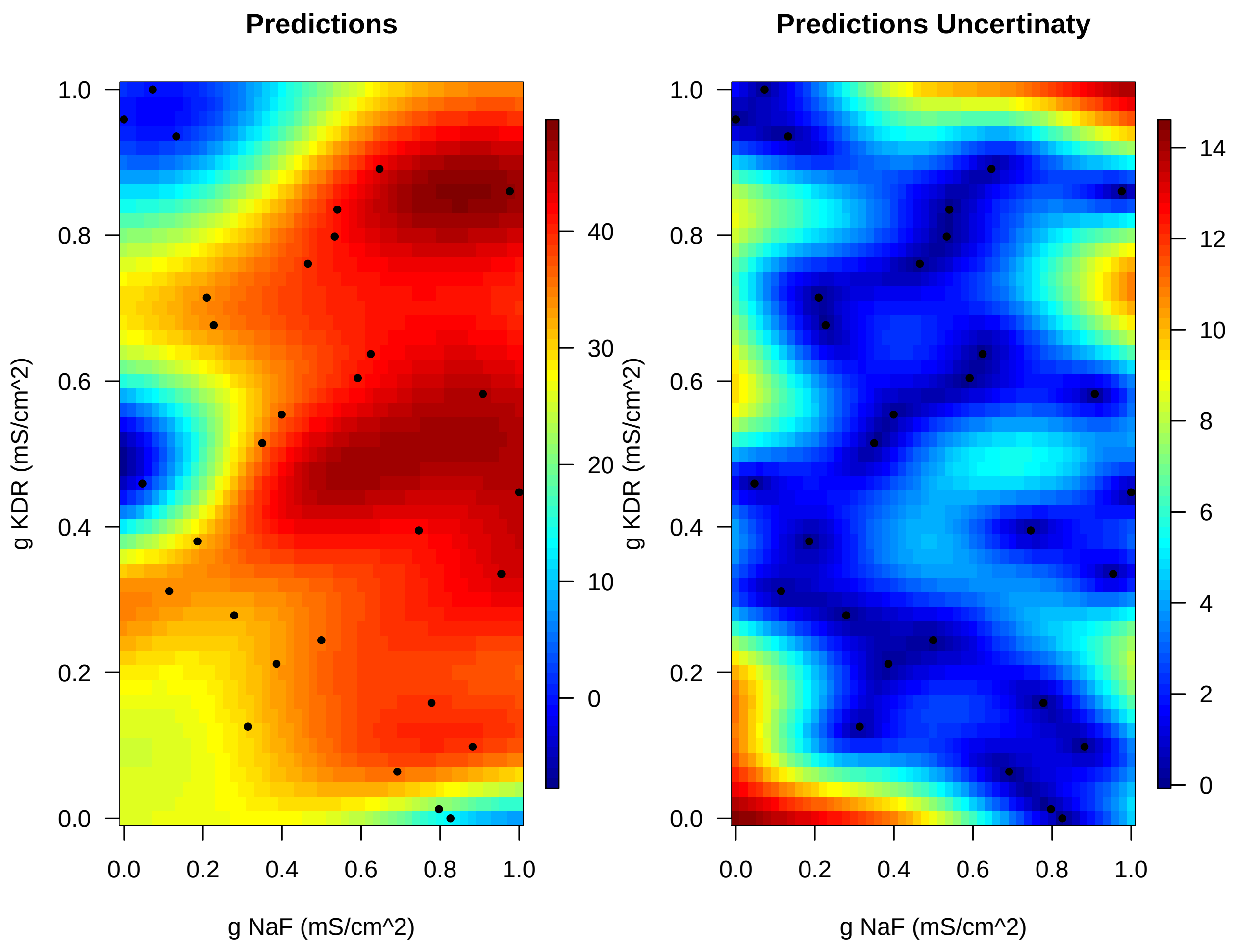
```
##
```

```
## Total observations = 30
## Dimensions = 2
##
## mu = 27.61157
## sig2:    251.8751
## nugget:  0
##
## Correlation parameters:
##
##        beta a
## 1  5.027878 2
## 2 50.228477 2
##
## Log likelihood = -104.4465
##
## CV RMSE: 7.312618
## CV RMaxSE: 1020.777
```

```
predictedX = expand.grid(g_NaF = seq(0, 1, 0.02), g_KDR = seq(0, 1, 0.02))
yhats = predict(GPFit, predictedX, se.fit = T)

library(fields)
```

```
## Loading required package: spam

## Loading required package: dotCall64

## Loading required package: grid

## Spam version 2.4-0 (2019-11-01) is loaded.
## Type 'help( Spam)' or 'demo( spam)' for a short introduction
## and overview of this package.
## Help for individual functions is also obtained by adding the
## suffix '.spam' to the function name, e.g. 'help( chol.spam)'.

##
## Attaching package: 'spam'

## The following objects are masked from 'package:base':
##
##     backsolve, forwardsolve

## Loading required package: maps

## See https://github.com/NCAR/Fields for
##  an extensive vignette, other supplements and source code
```

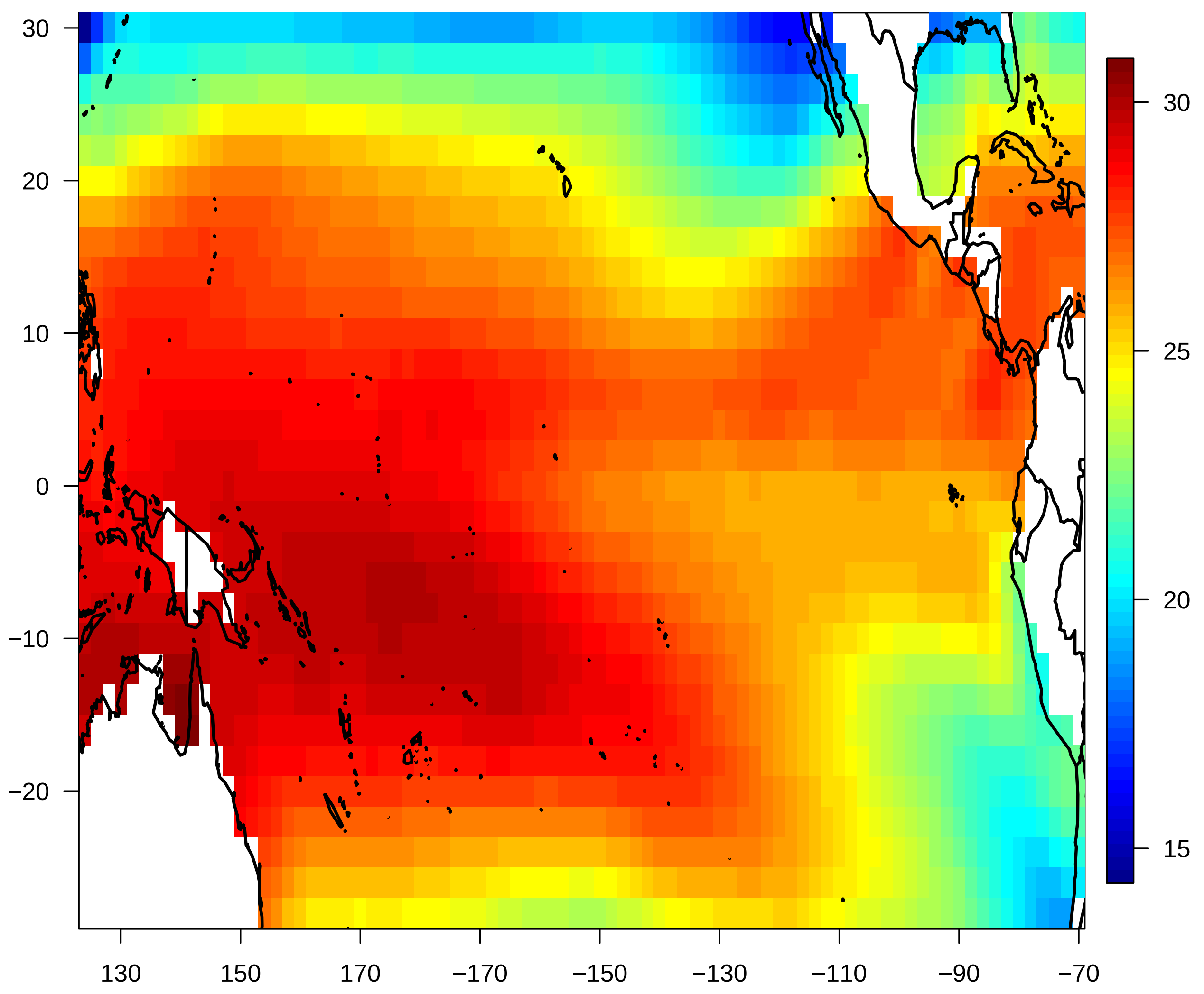
```
par(mfrow = c(1, 2))
image.plot(seq(0, 1, 0.02), seq(0, 1, 0.02), matrix(yhats$fit, 51, 51),
           xlab = "g NaF (mS/cm^2)", ylab = "g KDR (mS/cm^2)", las = 1,
           main = "Predictions")
points(neuron[, 1:2], pch = 16, cex = 0.75)
image.plot(seq(0, 1, 0.02), seq(0, 1, 0.02), matrix(yhats$se.fit, 51, 51),
           xlab = "g NaF (mS/cm^2)", ylab = "g KDR (mS/cm^2)", las = 1,
           main = "Predictions Uncertinaty")
points(neuron[, 1:2], pch = 16, cex = 0.75)
```



PCA: SST Example

Load and visualize the data

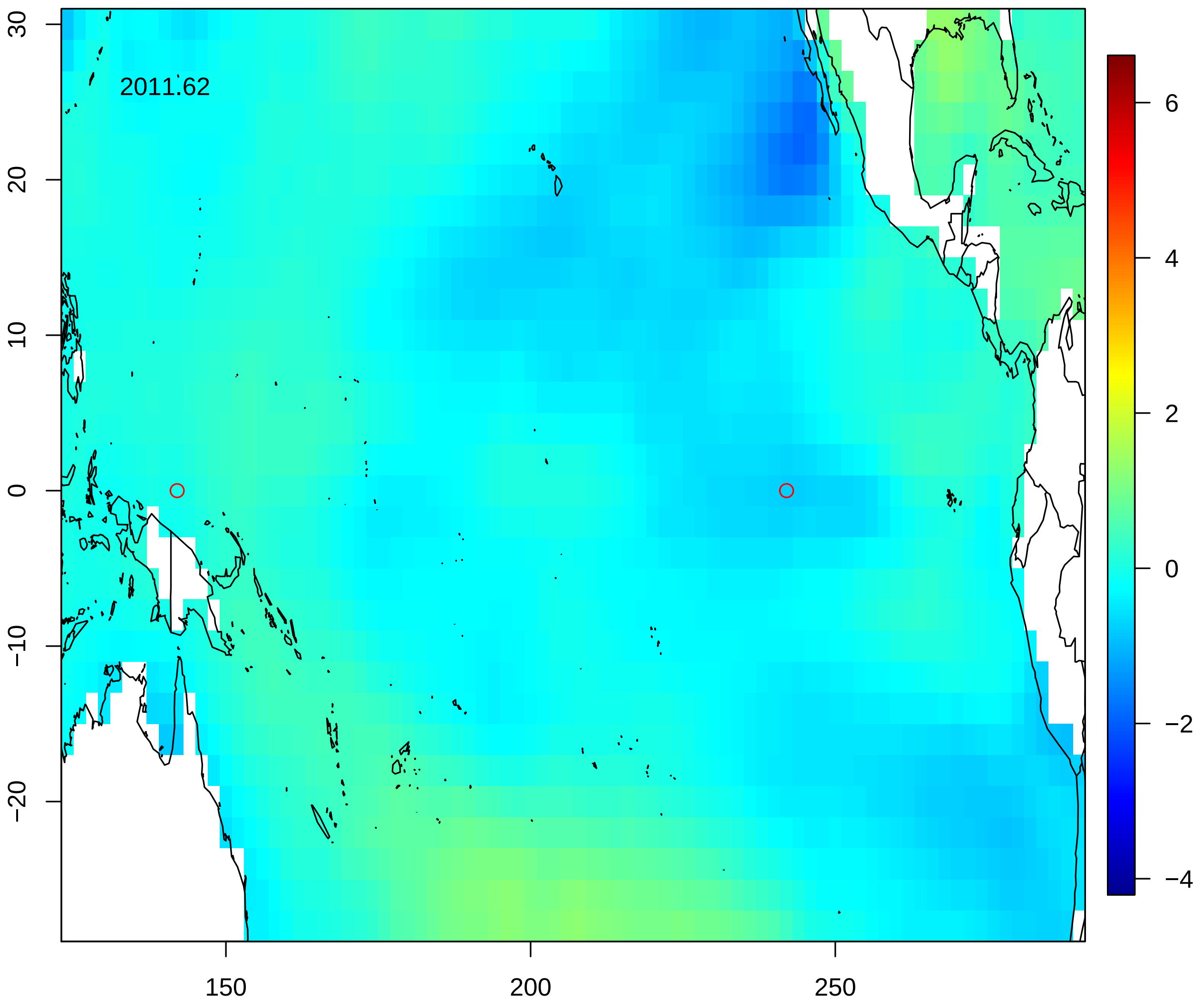
```
load("SST1.rda")
library(fields)
par(las = 1, mar = c(3, 3, 1, 1))
image.plot(lon1, lat1, SST1[,, 1],
           xaxt = "n", xlab = "", ylab = "")
lon <- ifelse(lon1 <= 180, lon1, lon1 - 360)
axis(1, at = lon1[seq(4, 84, 10)], lon[seq(4, 84, 10)])
map("world2", add = TRUE, lwd = 2)
```



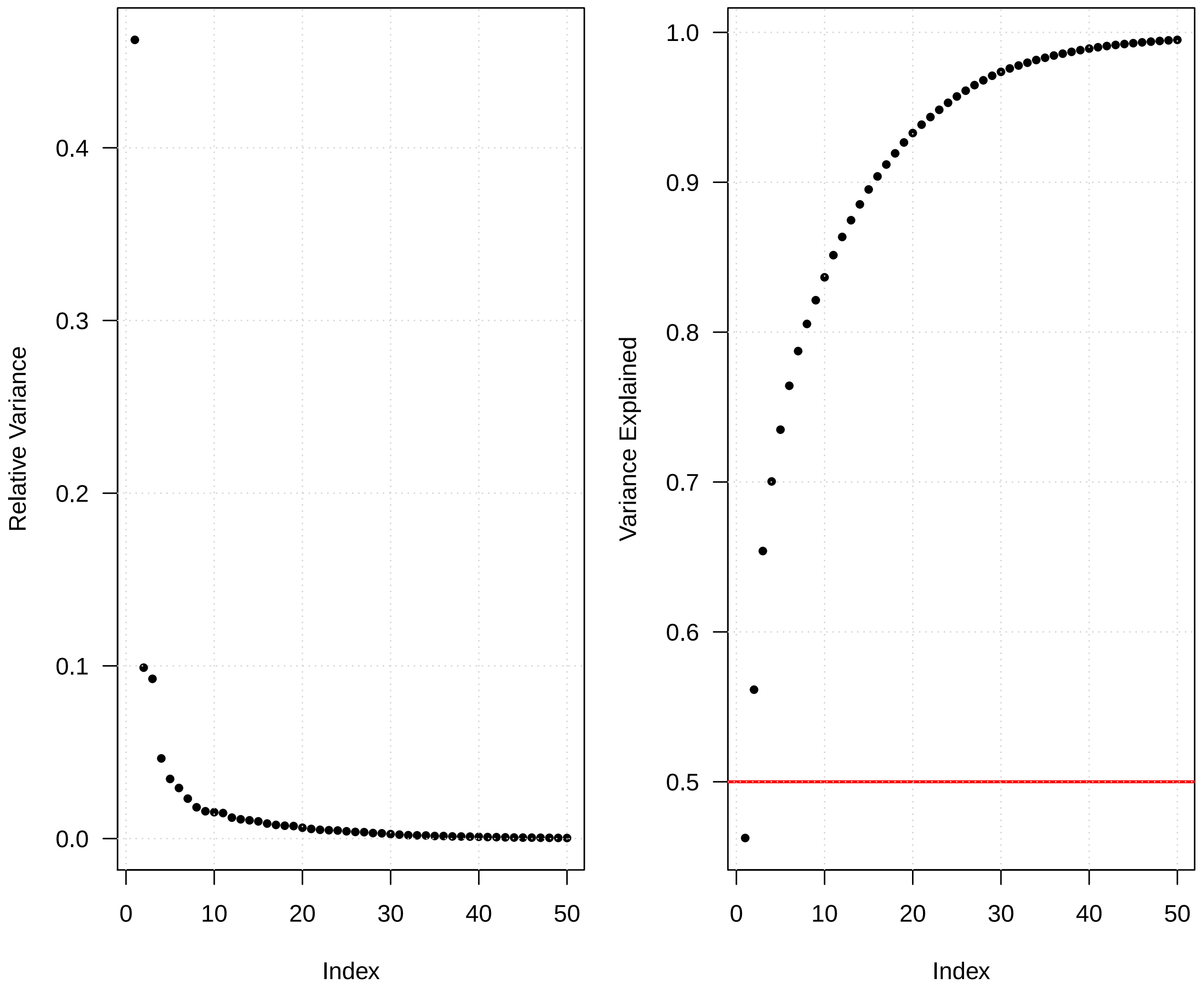
```r
t <- array(SST1, dim = c(84, 30, 12, 46))
SST_temp <- apply(t, 1:3, function(x) x - mean(x, na.rm = T))

SST_anomalies <- array(dim = c(84, 30, 552))
for (i in 1:84){
  for (j in 1:30){
    SST_anomalies[i, j,] <- c(t(SST_temp[, i, j,]))
  }
}
```

```r
SST <- SST_anomalies
k = 500
par(mar = c(3, 3, 1, 1))
zr <- range(SST, na.rm = TRUE)
image.plot(lon1, lat1, SST[, , k], zlim = zr, col = tim.colors(256),
           axes = TRUE, xlab = "", ylab = "")
map("world2", add = TRUE)
text(140, 26, labels = format(round(tm1[k], 2)), cex = 1)
ix <- c(10, 60)
iy <- c(15, 15)
points(lon1[ix], lat1[iy], pch = 1, cex = 1.2, col = "red")
```

```
box()
```



```
temp <- array(SST, c(84 * 30, 552))
ind <- is.na(temp[, 1])
temp <- temp[!ind,]
temp2 <- svd(temp)
U1 <- matrix(NA, 84 * 30)
U1[!ind] <- temp2$u[, 1]
U1 <- matrix(U1, 84, 30)

par(mar = c(4, 4, 1, 1), mfrow = c(1, 2), las = 1)
dt <- ((temp2$d^2) / sum(temp2$d^2))
plot(1:50, dt[1:50], xlab = "Index", ylab = "Relative Variance",
     pch = 16, cex = 0.8)
grid()
dt <- (cumsum(temp2$d^2)/sum(temp2$d^2))
plot(1:50, dt[1:50], xlab = "Index", ylab = "Variance Explained",
     pch = 16, cex = 0.8)
yline(0.5, col = "red", lwd = 2)
grid()
```



```
temp <- array(SST, c(84 * 30, 552))
ind <- is.na(temp[, 1])
temp <- temp[!ind, ]
temp2 <- svd(temp)
U1 <- matrix(NA, 84 * 30)
U1[!ind] <- temp2$u[, 1]
U1 <- matrix(U1, 84, 30)
U2 <- matrix(NA, 84 * 30)
U2[!ind] <- temp2$u[, 2]
U2 <- matrix(U2, 84, 30)
U3 <- matrix(NA, 84 * 30)
U3[!ind] <- temp2$u[, 3]
U3 <- matrix(U3, 84, 30)
zr <- range(c(U1, U2, U3), na.rm = TRUE)
#pdf("EOF123.pdf", 5, 5.5)
set.panel(3, 1)
```

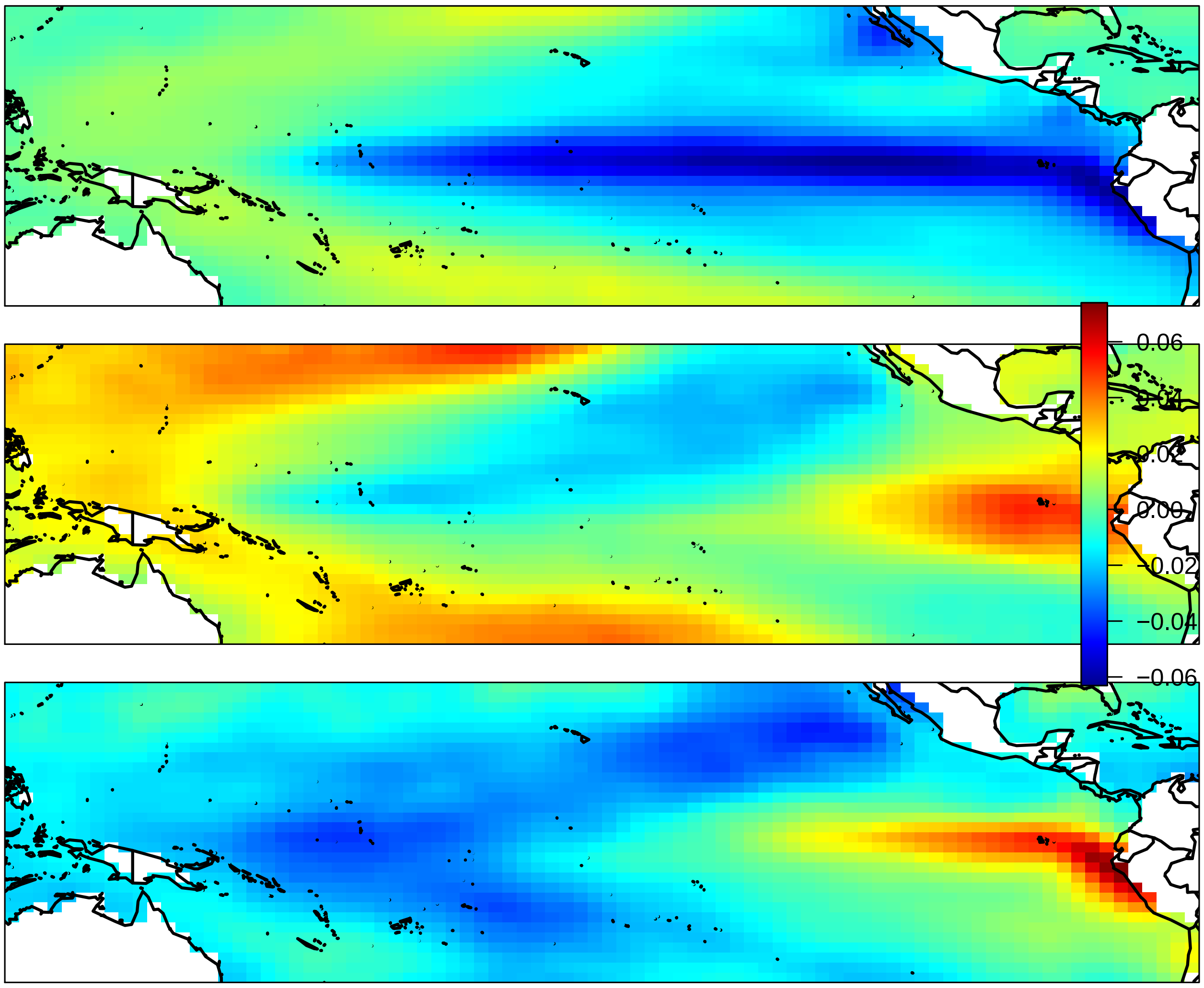
```
## plot window will lay out plots in a 3 by 1 matrix
```

```
par(oma = c(0, 0, 0, 0))
ct <- tim.colors(256)
par(mar = c(1, 1, 1, 1))
```

```
image(lon1, lat1, U1, axes = FALSE, xlab = "", ylab = "", zlim = zr,
      col = ct)
map("world2", add=TRUE, lwd=2)
box()
image(lon1, lat1, U2, axes = FALSE, xlab = "", ylab = "", zlim = zr,
      col = ct)
map("world2", add=TRUE, lwd=2)
box()
image(lon1, lat1, U3, axes = FALSE, xlab = "", ylab = "", zlim = zr,
      col = ct)
map("world2", add=TRUE, lwd=2)
box()

#dev.off()
set.panel()
```

```
## plot window will lay out plots in a 1 by 1 matrix
```

```
par(oma = c(0, 0, 0, 0))
image.plot(legend.only = TRUE, zlim = zr, col = ct, legend.shrink = 0.4)
```



```
#dev.off()
```

“‘