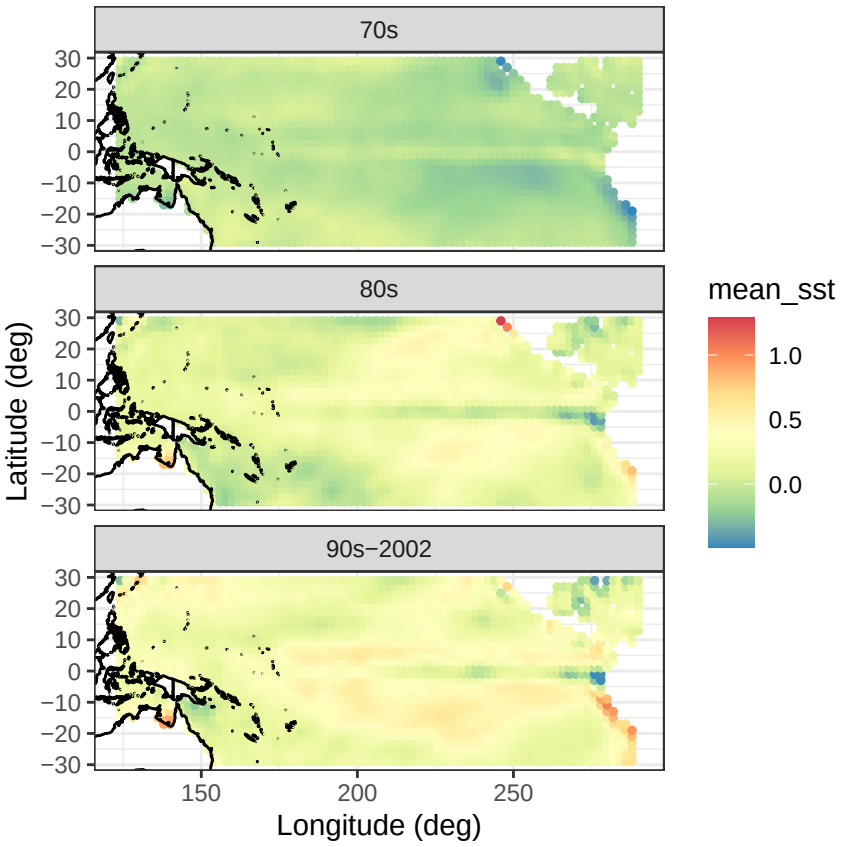
Dacade Means

For the SST data loaded using the code below: Generate a data frame with the Empirical Spatial Means per decade (1970-1979, 1980-1989, 1990-2002) and plot them with one panel per decade

```
# summary(SST_df)
SST_df <- SST_df %>% filter(!is.na(Year),!is.na(sst))
SST_df$decade <- cut(SST_df$Year, # bin the year into
                         c(1970,1980,1990,2002), # their respective bins
                         labels = c("70s","80s","90s-2002"), # assign labels
                         include.lowest = T) # include edges,right=F
summ <- SST_df %>% group_by(decade) %>% summarise(decade_means=mean(sst))
# year-year%%10 # floor_date(date,unit = 'year') # floor(decimal_date(date)/10)*10
head(summ)
## # A tibble: 3 x 2
##   decade   decade_means
##   <fct>           <dbl>
## 1 70s           -0.0722
## 2 80s            0.176
## 3 90s-2002       0.296
```

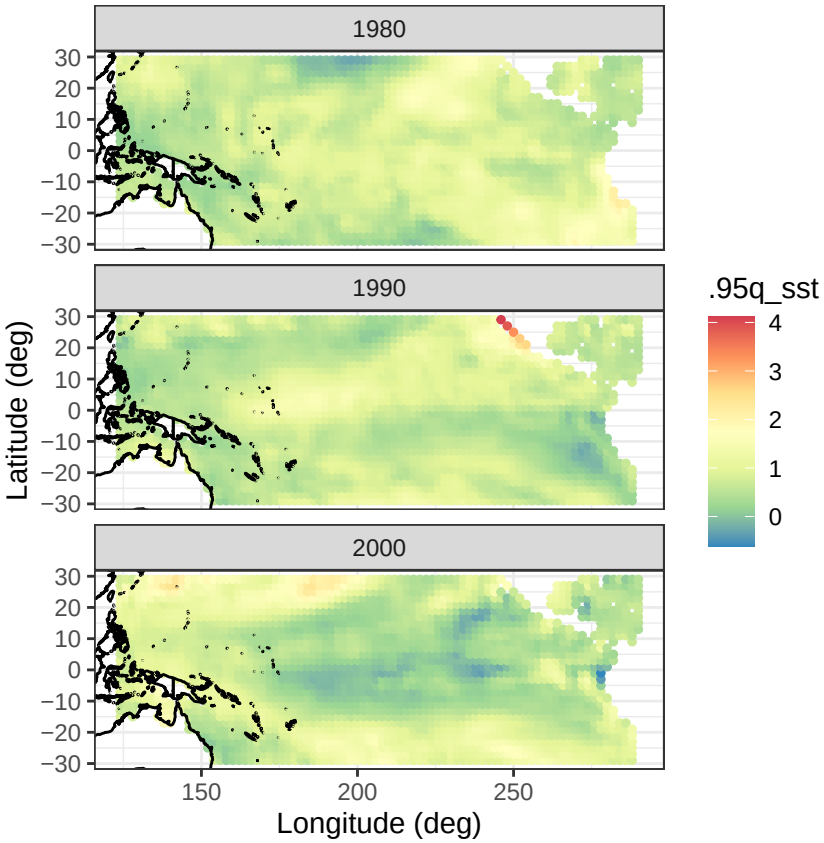
```
SST_df %>% group_by(lon,lat,decade) %>% summarise(z=mean(sst)) %>%
  ggplot() +
  geom_point(aes(x=lon,y=lat,colour=z),size=1) +
  col_scale(name = "mean_sst") +
  xlab("Longitude (deg)") + ylab("Latitude (deg)") +
  geom_path(data = map_data("world"),# world boundaries
            aes(x = long, y = lat, group = group)) +
  coord_fixed(xlim = c(124, 290),ylim= c(-29,29)) +
  facet_wrap(.~decade,nrow=3)+
  theme_bw()
```



yearly 95th quantile

generate a spatial plot for the **yearly** SST 95th quantile for the years 1980, 1990, 2000 having one panel per year

```
SST_df %>% filter(Year%in%c(1980,1990,2000)) %>%
  group_by(lon,lat,Year) %>% summarise(z=quantile(sst,.95)) %>%
  ggplot() +
  geom_point(aes(x=lon,y=lat,colour=z),size=1) +
  col_scale(name = ".95q_sst") +
  xlab("Longitude (deg)") + ylab("Latitude (deg)") +
  geom_path(data = map_data("world"),# world boundaries
            aes(x = long, y = lat, group = group)) +
  coord_fixed(xlim = c(124, 290),ylim= c(-29,29)) +
  facet_wrap(.~Year,nrow=3)+
  theme_bw()
```



Hovmoller plot

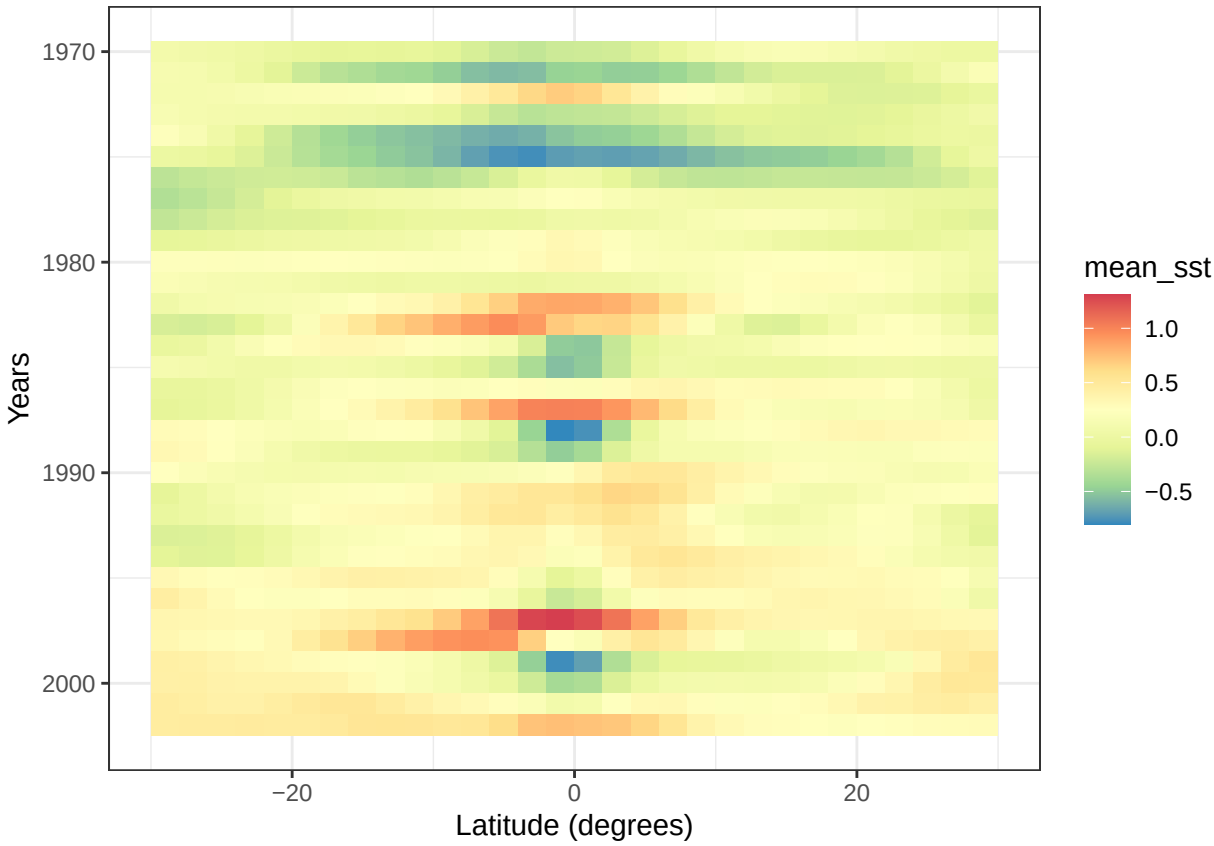
Obtain a Hovmoller plot for these data

```
lim_lat <- range(SST_df$lat)  # latitude range
lim_t <- range(SST_df$Year) # time range
lat_axis <- seq(lim_lat[1], lim_lat[2],length=59) # latitude axis
t_axis <- seq(lim_t[1], lim_t[2],length=33)# time axis
lat_t_grid <- expand.grid(lat = lat_axis, t = t_axis)
```

```
SST_grid <- SST_df
dists <- abs(outer(SST_df$lat, lat_axis, "-"))
SST_grid$lat <- lat_axis[apply(dists, 1, which.min)]
```

```
SST_lat_Hov <- SST_grid %>% group_by(lat, Year) %>%
summarise(z = mean(sst))
```

```
ggplot(SST_lat_Hov) +
  geom_tile(aes(x = lat, y = Year, fill = z)) +
  fill_scale(name = "mean_sst") + scale_y_reverse() +
  ylab("Years") + xlab("Latitude (degrees)") +
  theme_bw()
```



EOFs

Calculate the EOFs for the *SST* dataset. Replicate the figures in pages 44-45 using the R function **eigen** (not **svd**) and ggplot2::geom_tile and interpret them. How many EOFs would you retain?

```
Z <- t(SSTdata) # dim(Z)
spat_mean <- apply(SSTdata, 1, mean)
nT <- ncol(SSTdata)
nS <- nrow(SSTdata)
Zspat_detrend <- Z - outer(rep(1, nT), spat_mean) #subtract and standardize:
Zt <- 1/sqrt(nT - 1)*Zspat_detrend # dim(Zt)
```

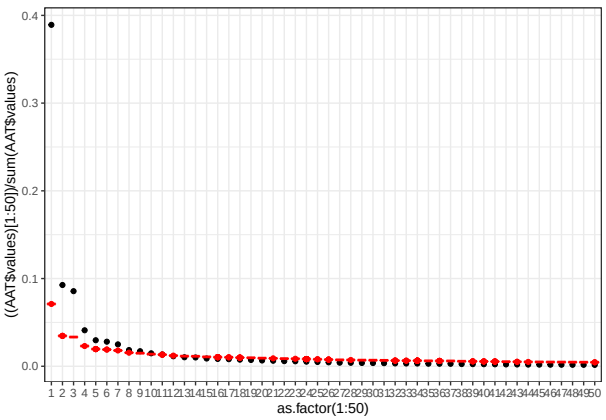
```
ATA <- eigen(t(Zt)%*%Zt,symmetric=T)
V <- ATA$vectors[,1:nT]
AAT <- eigen(Zt%*%t(Zt),symmetric=T)
U <- AAT$vectors
D <- diag(sqrt(AAT$values))
```

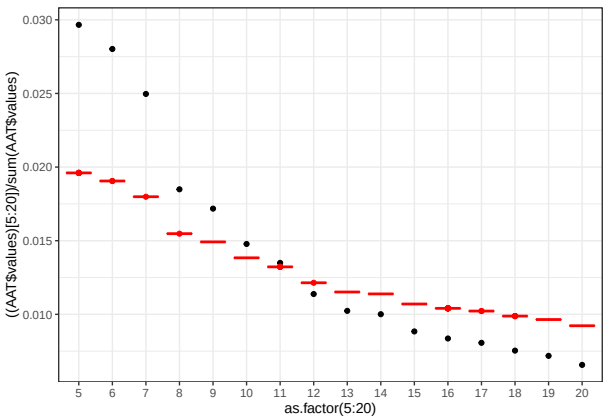
```
round(E$v[1:5,1:8],4)
##         [,1]    [,2]    [,3]   [,4]   [,5]    [,6]    [,7]    [,8]
## [1,] -0.0049 -0.0121 -0.0288 0.0001 0.0154 -0.0380 -0.0160  0.0053
## [2,] -0.0014 -0.0023 -0.0255 0.0067 0.0184 -0.0372 -0.0188 -0.0139
## [3,]  0.0002  0.0023 -0.0193 0.0086 0.0204 -0.0406 -0.0204 -0.0172
## [4,]  0.0015  0.0023 -0.0191 0.0103 0.0200 -0.0406 -0.0181 -0.0160
## [5,]  0.0023  0.0016 -0.0225 0.0113 0.0208 -0.0431 -0.0170 -0.0171
round(V[1:5,1:8],4)
##          [,1]    [,2]    [,3]    [,4]   [,5]    [,6]    [,7]    [,8]
## [1,]  0.0049  0.0121 -0.0288 -0.0001 0.0154 -0.0380 -0.0160  0.0053
## [2,]  0.0014  0.0023 -0.0255 -0.0067 0.0184 -0.0372 -0.0188 -0.0139
## [3,] -0.0002 -0.0023 -0.0193 -0.0086 0.0204 -0.0406 -0.0204 -0.0172
## [4,] -0.0015 -0.0023 -0.0191 -0.0103 0.0200 -0.0406 -0.0181 -0.0160
## [5,] -0.0023 -0.0016 -0.0225 -0.0113 0.0208 -0.0431 -0.0170 -0.0171
V[,c(1,2,4)] <- V[,c(1,2,4)]*-1
```

Use 'eigen' function, the columns 1, 2, 4 in "V" matrix have inverse sign to these in "V" by "svd" function

```
colnames(V) <- paste0("EOF", 1:ncol(SSTdata)) # label columns
EOFs <- cbind(SSTlonlat[-rm_rows, ],V) # head(EOFs[,1:6])
```

```
E100 <- matrix(rep(NA,100*nT),100,nT) %>% as.data.frame()
for (i in 1:100) {
s.id <- sample(1:nS,nS,replace = F)
E100[i,] <- svd(Zt[,s.id])$d
}
colnames(E100) <-1:nT  #paste0("rank", 1:nT)
E100.long <- E100[,1:50] %>%
      gather(rank,EOF,factor_key=T)
E100.long$EOF <- E100.long$EOF/sum(E$d)
```

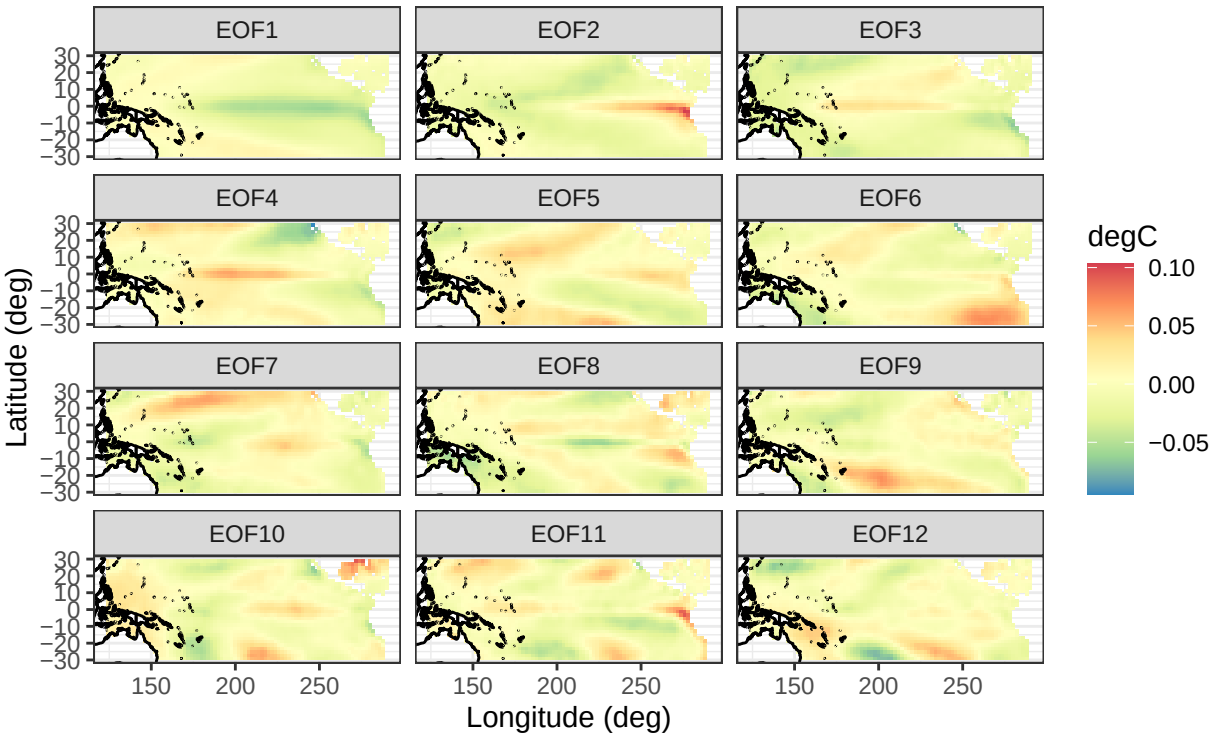




The black points in the scree plot correspond to the relative variance associated with the first 50 EOFs for the SST data. The red boxplots of relative variances obtained from EOF analyses of 100 random permutations of the data.

The zoom-in plot shows the two curves intersect between 11 and 12, suggesting that there is very little "real" variability being accounted for by the EOFs with indices greater than about 12.

```
EOFs.long <- EOFs[,1:14] %>%
      gather(EOF_id,EOF ,-lon,-lat,factor_key=T) # levels(EOFs.long$EOF_id) <- 1:12
ggplot(EOFs.long) + geom_tile(aes(x = lon, y = lat, fill =EOF)) + #
   fill_scale(name = "degC") +
   xlab("Longitude (deg)") + ylab("Latitude (deg)")+
  geom_path(data = map_data("world"),# world boundaries
            aes(x = long, y = lat, group = group)) +
  coord_fixed(xlim = c(124, 290),ylim= c(-29,29)) +
  facet_wrap(vars(EOF_id),ncol = 3) + # ,labeller = "label_value"
  theme_bw()
```



https://stats.idre.ucla.edu/r/codefragments/svd_demos/

inclass problems 1

- Semivariogram

Let Z_1, Z_2 represent $Z(s+h;t+\tau)$ and $Z(s;t)$, μ_1, μ_2 represent $\mu(s+h;t+\tau)$ and $Z(s;t)$,

$$\begin{aligned} C_z(h;\tau) &= E[(Z(s;t)-\mu(s;t))(Z(s+h;t+\tau)-\mu(s+h;t+\tau)] \\ &= E[(Z_1-\mu_1)(Z_2-\mu_2)] \\ &= \mathrm{Cov}[Z_1, Z_2] \end{aligned}$$

$$\begin{aligned} C_z(0;0) &= E[(Z(s+h;t+\tau)-\mu(s+h;t+\tau)^2] = \mathrm{Var}[Z_1] \\ C_z(0;0) &= E[(Z(s;t)-\mu(s;t))^2] = \mathrm{Var}[Z_2] \end{aligned}$$

$$\begin{aligned} \mathrm{Var}[Z_1 - Z_2] &= \mathrm{Var}[Z_1] + \mathrm{Var}[Z_2] - 2\mathrm{Cov}[Z_1, Z_2] \\ &= C_z(0,0) + C_z(0,0) - 2C_z(h,\tau) \end{aligned}$$

Therefore,

$$\frac{1}{2}\mathrm{Var}[Z(s+h;t+\tau) - Z(s;t)] = C_z(0,0) - C_z(h,\tau).$$

- Canonical correlation

$$\begin{aligned} a_k(t_j) &= \sum_{i=1}^{m} \xi_{ik} Z(s_i;t_j) \;=\; \boldsymbol{\xi}_k' \mathbf{Z}_{t_j} \\ b_k(t_j) &= \sum_{\ell=1}^{n} \psi_{\ell k} X(r_\ell;t_j) \;=\; \boldsymbol{\psi}_k' \mathbf{X}_{t_j} \end{aligned}$$

$$\begin{aligned} Var[a_k(t_j)] &= \boldsymbol{\xi}_k' \mathbf{Z}_{t_j} (\boldsymbol{\xi}_k' \mathbf{Z}_{t_j})' = \boldsymbol{\xi}_k' \mathbf{Z}_{t_j} \mathbf{Z}_{t_j}' \boldsymbol{\xi}_k = \boldsymbol{\xi}_k' C_z^{(0)} \boldsymbol{\xi}_k \quad (C_z \text{ is symmetric}) \\ Var[b_k(t_j)] &= \boldsymbol{\psi}_k' \mathbf{X}_{t_j} (\boldsymbol{\psi}_k' \mathbf{X}_{t_j})' = \boldsymbol{\psi}_k' \mathbf{X}_{t_j} \mathbf{X}_{t_j}' \boldsymbol{\psi}_k = \boldsymbol{\psi}_k' C_x^{(0)} \boldsymbol{\psi}_k \quad (C_x \text{ is symmetric}) \\ Cov[a_k(t_j), b_k(t_j)] &= \boldsymbol{\xi}_k' \mathbf{Z}_{t_j} (\boldsymbol{\psi}_k' \mathbf{X}_{t_j})' = \boldsymbol{\xi}_k' \mathbf{Z}_{t_j} \mathbf{X}_{t_j}' \boldsymbol{\psi}_k = \boldsymbol{\xi}_k' C_{z,x}^{(0)} \boldsymbol{\psi}_k \end{aligned}$$

$$r_k = \mathrm{corr}(a_k, b_k) = \frac{Cov[a_k(t_j), b_k(t_j)]}{\sqrt{Var[a_k(t_j), Var[b_k(t_j)]]}} = \frac{\boldsymbol{\xi}_k' \mathbf{C}_{z,x}^{(0)} \boldsymbol{\psi}_k}{(\boldsymbol{\xi}_k' \mathbf{C}_z^{(0)} \boldsymbol{\xi}_k)^{1/2} (\boldsymbol{\psi}_k' \mathbf{C}_x^{(0)} \boldsymbol{\psi}_k)^{1/2}}$$

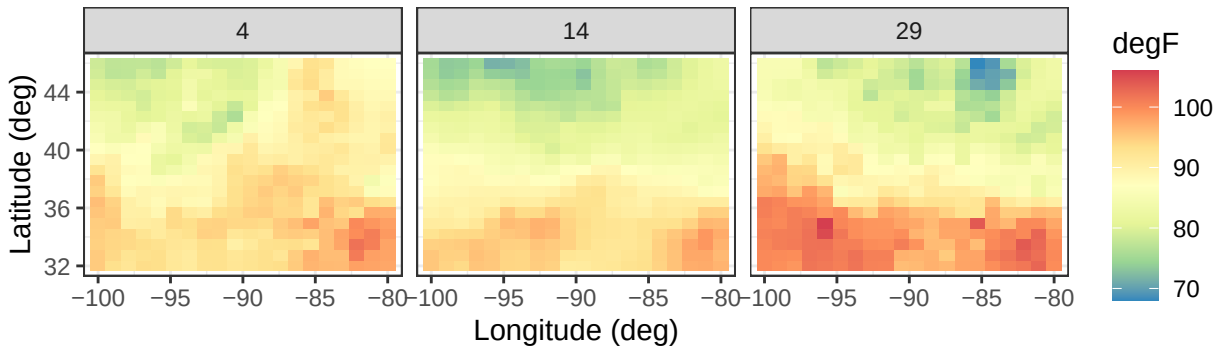
inclass problems 2

```
data("NOAA_df_1990", package = "STRbook")
Tmin <- filter(NOAA_df_1990,       # subset the data
               proc == "Tmin" &    # only min temperature
               month %in% 7 &      # only July
               year == 1993 &     # year of 1993
               day != 14) %>%     # remove day 14
               mutate(t=as.integer(julian-min(julian-1))) #create time variable
pred_grid <- expand.grid(lon = seq(-100, -80, length = 20),
                         lat = seq(32, 46, length = 20),
                         day = c(4, 14, 29))
```

```
Tmin_July_idw <- idw(formula = z ~ 1,            # dep. variable
                     locations = ~ lon + lat + day, # inputs
                     data = Tmin,                # data set
                     newdata = pred_grid,        # prediction grid
                     idp = 5)                    # inv. dist. pow.
## [inverse distance weighted interpolation]
```

```
pred_obs_dist_mat <- rdist(select(pred_grid, lon, lat, day),
                           select(Tmin, lon, lat, day))
Wt_IDW <- function(theta, dist_mat) 1/dist_mat^theta
Wtilde <- Wt_IDW(theta = 5, dist_mat = pred_obs_dist_mat)
Wtilde_rsums <- rowSums(Wtilde)
W <- Wtilde/Wtilde_rsums # Weights
z_pred_IDW <- as.numeric(W %*% Tmin$z)
pred_grid$z.hat <- as.numeric(W %*% Tmin$z)
```

```
ggplot(pred_grid) +
    geom_tile(aes(x = lon, y = lat,fill = z.hat)) +
    fill_scale(name = "degF") +                          # attach color scale
    xlab("Longitude (deg)") + ylab("Latitude (deg)") +   # x,y-axis label
    facet_wrap(~ day, ncol = 3) +                        # facet by day
    coord_fixed(xlim = c(-100, -80),ylim = c(32, 46))  + theme_bw()
```




```
summary(Tmin_July_idw$var1.pred - z_pred_IDW)
##       Min.    1st Qu.     Median       Mean    3rd Qu.       Max.
## -1.535e-12 -1.705e-13  0.000e+00 -2.475e-15  1.705e-13  1.009e-12
```

The results of computed predictions between 'idw' function and using 'rdist' are very close.

inclass problems 3

Using the dataset

```
Tmax_long <- NOAA_df_1990 %>% # now subset the data
filter(proc == "Tmax" & # only max temperature
month %in% 7 & # only July
year == 1993) %>%
mutate(t=as.integer(julian-min(julian-1))) #create time variable
```

together with the functions defined below,cross-validate predictions

```
library(fields)
# "data" must include variables: lon, lat, t and z
K.fold.cv <- function(data,nfolds=5,weight.fn,theta){
mT <- nrow(data)
Z <- data$z
coords <- data %>% dplyr::select(lon,lat,t)
dist_mat <- rdist(coords,coords)
# sample fold label vector
if(nfolds < mT){fold.vec <- sample(1:nfolds,mT,replace = T)
          }else{fold.vec <- 1:mT}
w.tilde <- weight.fn(theta,dist_mat)
MSPEk <- 1:nfolds %>%
         map_dbl(function(x){
                 hold.out <- which(fold.vec==x)
                        w <- w.tilde[hold.out,-hold.out,drop=F]
                        w <- w * (1/rowSums(w))
                    Z.hat <- w %*% Z[-hold.out]
                    mean((Z[hold.out]-Z.hat)^2)
                }) # CV score
return(mean(MSPEk))
}
```

with a Gaussian kernel

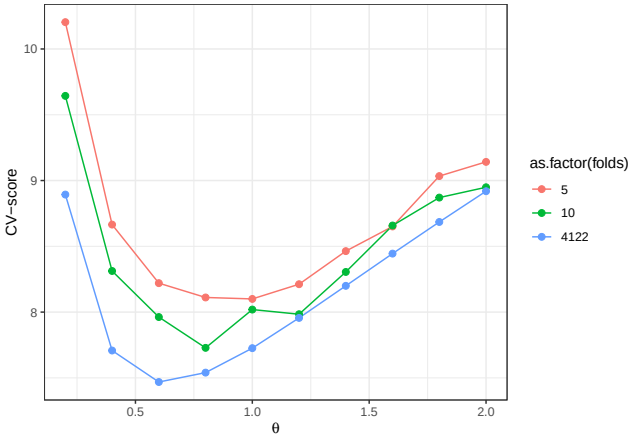
```
weight.gauss <- function(theta, dist_mat){
                          exp(-dist_mat^2/theta)
}
```

setting the number of folds to $K = 5, 10, m_T$(leave-one-outCV), and the values of $\theta = 0.2, 0.4, ..., 2$.

```
theta <- seq(0.2,2,0.2)
folds <- c(5,10,nrow(Tmax_long))
cv <- data.frame(folds=rep(folds,each=length(theta)),
           theta=rep(theta,times=length(folds)),
           score=NA)
for (i in 1:3) { #
  for (j in 1:10) { #
      cv[(i-1)*10+j,3] <- K.fold.cv(data=Tmax_long,
      nfolds=folds[i],
      weight.fn = weight.gauss,
      theta=theta[j])
```

```
  }
}
```

Compare the 5, 10 and LOO cv procedures by contrasting their corresponding CV-scores vs θ curves.




```
cv.map <- data.frame(folds=rep(folds,each=length(theta)),
           theta=rep(theta,times=length(folds)),
           score=NA)
for (i in 1:3) {
cv.map[(1:10)*i,3] <- theta %>% map_dbl(~K.fold.cv(data=Tmax_long,
      nfolds=folds[i],
      weight.fn = weight.gauss,
      theta=.x))
}
cv.map[10:30,]
##    folds theta    score
## 10     5   2.0 7.893370
## 11    10   0.2       NA
## 12    10   0.4 7.539927
## 13    10   0.6       NA
## 14    10   0.8 8.290413
## 15    10   1.0 7.725883
## 16    10   1.2 8.520115
## 17    10   1.4       NA
## 18    10   1.6 7.955813
## 19    10   1.8       NA
## 20    10   2.0 8.981705
## 21  4122   0.2 8.199697
## 22  4122   0.4       NA
## 23  4122   0.6       NA
## 24  4122   0.8 8.444584
## 25  4122   1.0       NA
## 26  4122   1.2       NA
## 27  4122   1.4 8.685022
## 28  4122   1.6       NA
## 29  4122   1.8       NA
## 30  4122   2.0 8.918847
```

However, using the same "k.fold.cv" and "weight.gauss" functions, there are some NA values.