Class Notes

Statistical Computing & Machine Learning

Class 19

Aside: Is continuity always good?

Regression discontinuity video

Quantiles

Show the ecdf() of some data. Invert this to get the quantiles.

Show `mosaic::qdata()` and `mosaic::pdata()` as inverses.

`quantile()` is the original version of this.

Standard errors and error bands

```
mod <- lm(time ~ year, data=SwimRecords)
summary(mod)
```

```
##
## Call:
## lm(formula = time ~ year, data = SwimRecords)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -9.8385 -4.2293  0.5861  3.2304 23.6035
##
## Coefficients:
##              Estimate Std. Error t value
## (Intercept) 567.24200   53.86572  10.531
## year         -0.25988    0.02759  -9.419
##             Pr(>|t|)
## (Intercept) 2.91e-15 ***
## year        1.96e-13 ***
## ---
## Signif. codes:
##   0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1  ' ' 1
##
## Residual standard error: 6.351 on 60 degrees of freedom
## Multiple R-squared:  0.5966, Adjusted R-squared:  0.5898
## F-statistic: 88.72 on 1 and 60 DF,  p-value: 1.958e-13
```

Do bootstrap replications

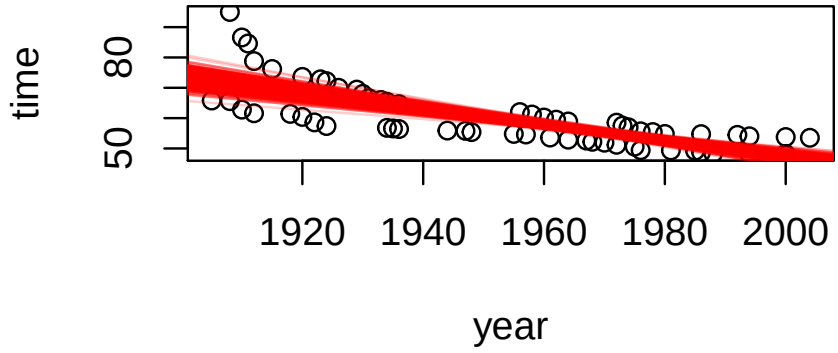
```
reps <- do(500) * lm(time ~ year,   data=resample(SwimRecords))
sd(reps$year)
```

```
## [1] 0.03290274
```

```
sd(reps$Intercept)
```

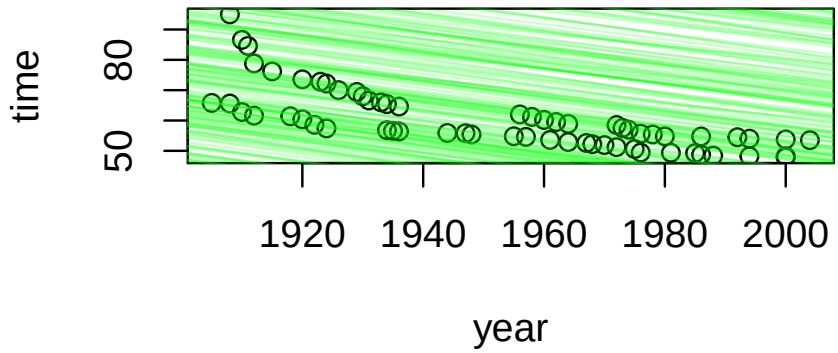
```
## [1] 64.77237
```

```
plot(time ~ year, data=SwimRecords)
for (k in 1:nrow(reps))
  abline(reps[k,]$Intercept, reps[k,]$year, col=rgb(1,0,0,.2))
```



Generate random values of slope and intercept and add those to the plot.

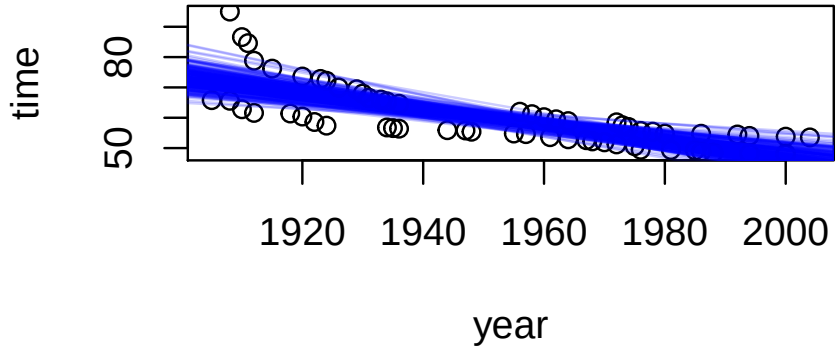
```
slopes <- rnorm(500, mean = -0.26, sd = 0.028)
intercepts <- rnorm(length(slopes), mean=567.2, sd=53.9)
if(!is.null(knitr::current_input() ))
   plot(time ~ year, data=SwimRecords)
for(k in 1:length(slopes))
  abline(intercepts[k], slopes[k], col=rgb(0,1,0, .2))
```



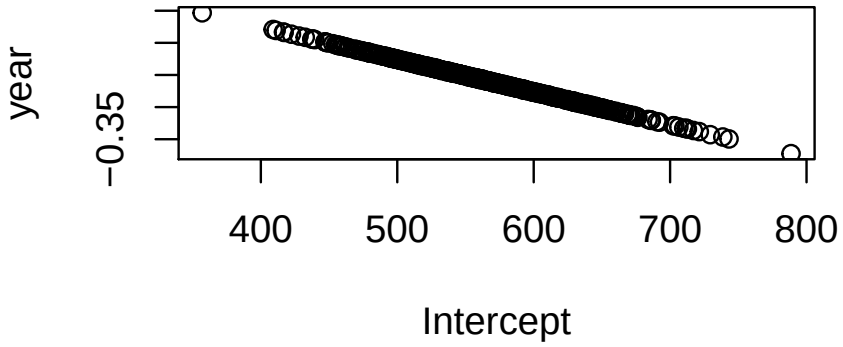
Why the disagreement between the bootstrapped coefficients and the random generation of slopes and intercepts?

```
rands <- matrix(rnorm(500, sd=2), ncol=2)
# 2 so that we cover 95% interval
```

```
better <- rands %*% chol(vcov(mod))
means <- matrix(c(567.2, -0.26), byrow = TRUE,ncol=2, nrow=nrow(better))
better <- better + means
if(!is.null(knitr::current_input() ))
   plot(time ~ year, data=SwimRecords)
for(k in 1:nrow(better))
  abline(better[k,1], better[k,2], col=rgb(0,0,1, .2))
```



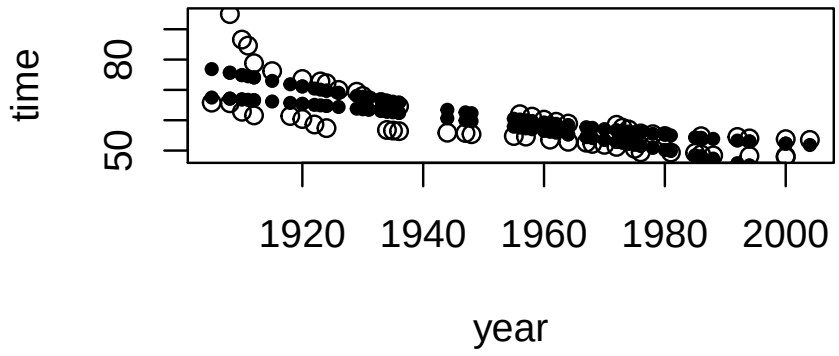

```
with(reps, plot(Intercept, year))
```



To generate standard error at a set of points:

```
rowSums(MM %*% vcov(mod) * MM)
# note standard multiplication *

MM <- model.matrix(time ~ year, data=SwimRecords)
SE <- rowSums(MM %*% vcov(mod) * MM)
top <- fitted(mod) + 2*SE
bottom <- fitted(mod) - 2*SE
if(!is.null(knitr::current_input() ))
   plot(time ~ year, data=SwimRecords)
points(MM[,2], top, pch=20)
points(MM[,2], bottom, pch=20)
```



Polynomial bases

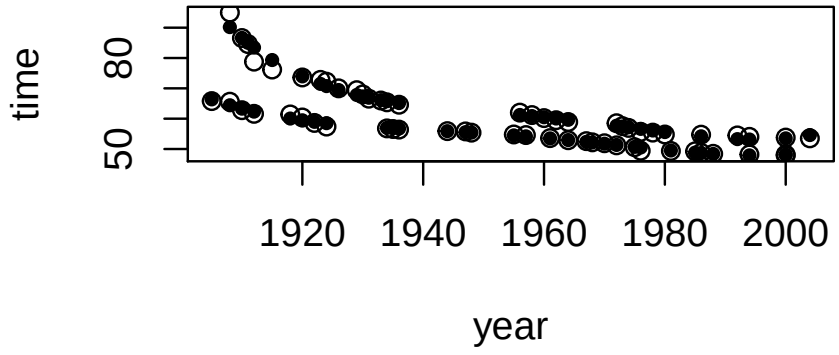
```
X <- data.frame(year = 1890:2020)
X$sex <- rep(c("F","M"), length=nrow(X))
```

The raw polynomials

```
x <- 0:100
funs <- outer(x, 0:5, FUN = '^')
svd(funs)$d
```

```
## [1] 3.098450e+10 3.407944e+07 1.068421e+05
## [4] 8.098882e+02 1.681222e+01 1.794720e+00
```

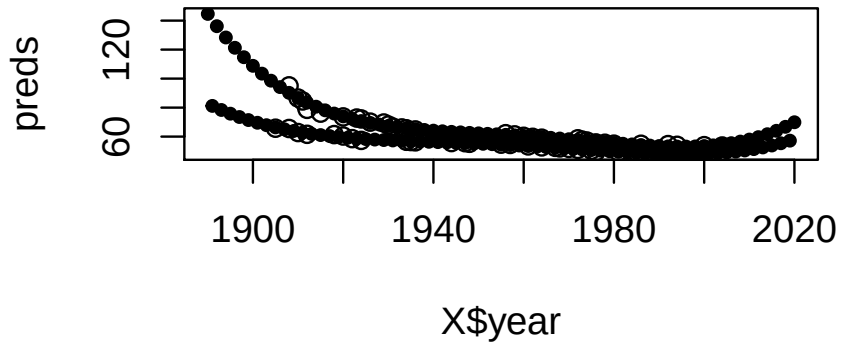
```
mod <- lm(time ~ sex*outer(year, 0:5, FUN = '^'), data=SwimRecords)
plot(time ~ year, data=SwimRecords)
with(SwimRecords, points(year, fitted(mod), pch=20))
```




```
preds <- predict(mod, newdata = X)
```

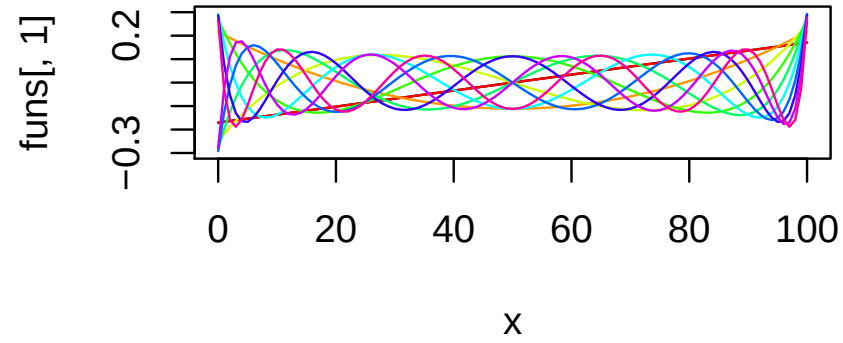
```
## Warning in predict.lm(mod, newdata = X):
## prediction from a rank-deficient fit may be
## misleading
```

```
plot(X$year, preds, pch=20)
with(SwimRecords, points(year, time))
```



Orthogonal polynomials

```
x <- 0:100
funs <- stats::poly(x, degree = 10)
plot(x, funs[,1], type="l", ylim=c(-.3,.3))
for (k in 1:ncol(funs))
  lines(x, funs[,k], col=rainbow(ncol(funs))[k])
```



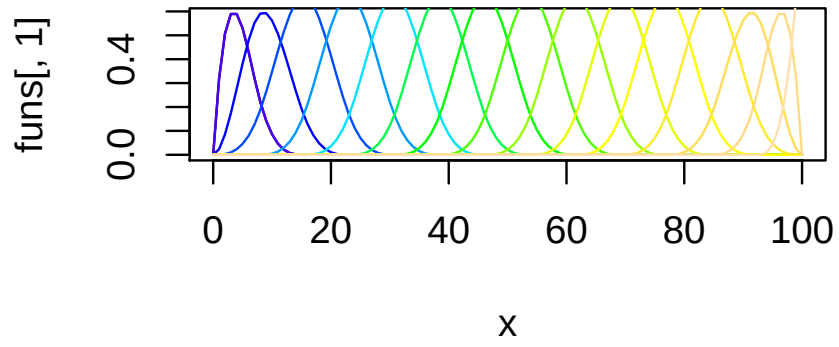

```
svd(funs)$d
```

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

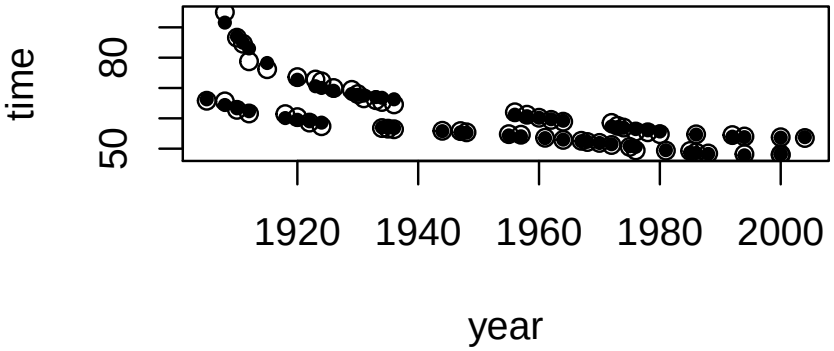
Splines

B-splines

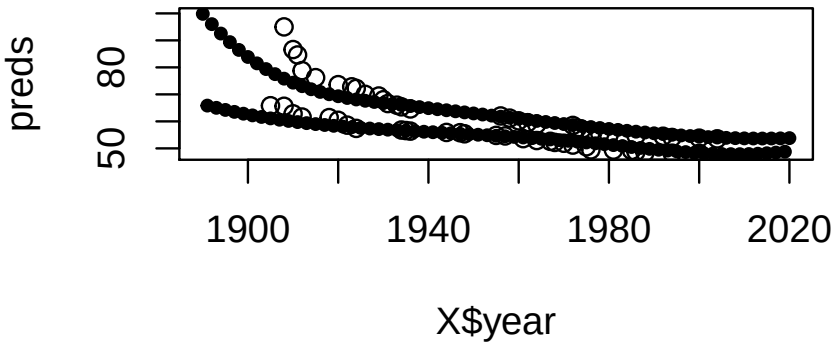
```
x <- 0:100
funs <- splines::bs(x, df = 15)
plot(x, funs[,1], type="l")
for (k in 1:ncol(funs))
  lines(x, funs[,k], col=topo.colors(ncol(funs))[k])
```




```
mod <- lm(time ~ sex * splines::bs(year, df=5), data=SwimRecords)
plot(time ~ year, data=SwimRecords)
with(SwimRecords, points(year, fitted(mod), pch=20))
```

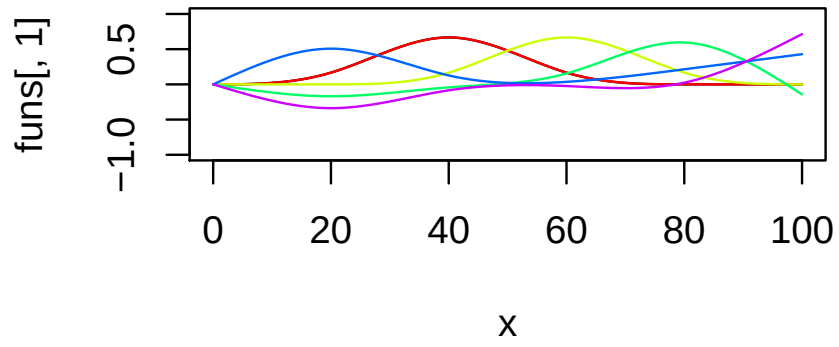



```
preds <- predict(mod, newdata = X)
plot(X$year, preds, pch=20)
with(SwimRecords, points(year, time))
```

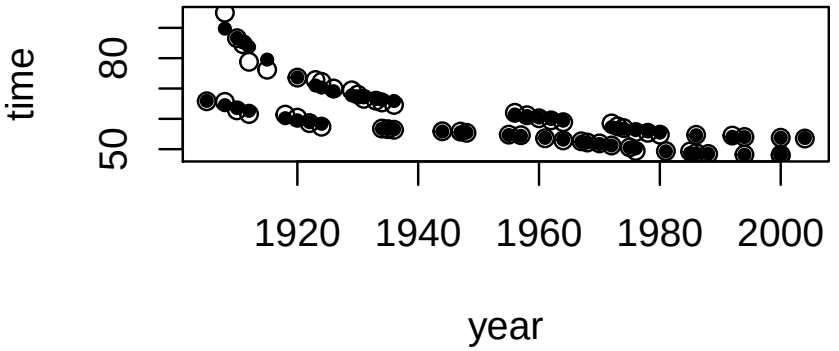


Natural splines

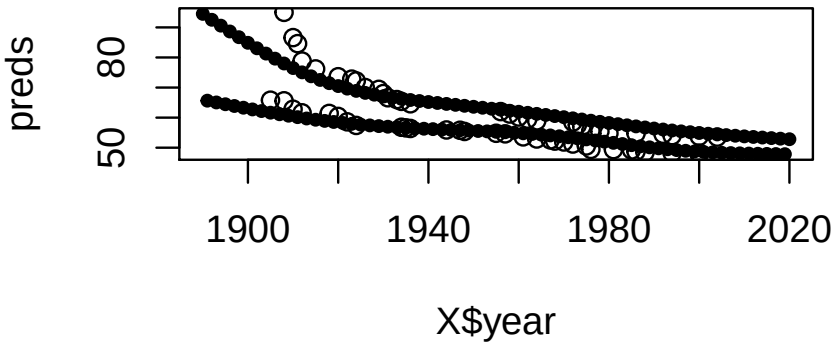
```
x <- 0:100
funs <- splines::ns(x, df = 5)
plot(x, funs[,1], type="l", ylim=c(-1,1))
for (k in 1:ncol(funs))
  lines(x, funs[,k], col=rainbow(ncol(funs))[k])
```




```
mod <- lm(time ~ sex * splines::ns(year, df=5), data=SwimRecords)
plot(time ~ year, data=SwimRecords)
with(SwimRecords, points(year, fitted(mod), pch=20))
```




```
preds <- predict(mod, newdata = X)
plot(X$year, preds, pch=20)
with(SwimRecords, points(year, time))
```



Smoothing splines

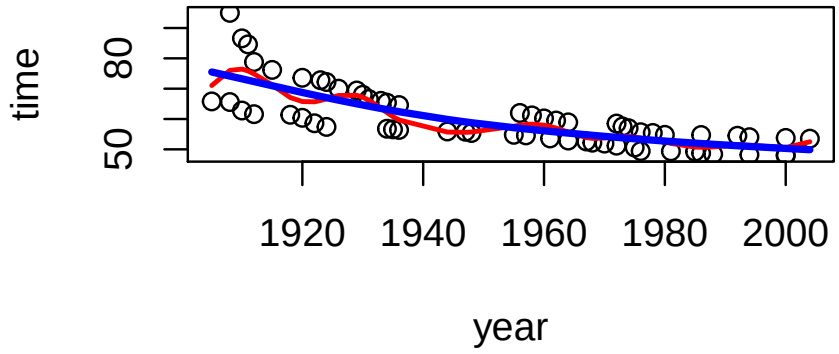
```
mod <- with(SwimRecords, smooth.spline(year, time, df = 15))
mod2 <- with(SwimRecords, smooth.spline(year, time, cv = TRUE))
```

```
## Warning in smooth.spline(year, time, cv =
## TRUE): cross-validation with non-unique 'x'
## values seems doubtful
```

```
mod2$df
```

```
## [1] 3.47454
```

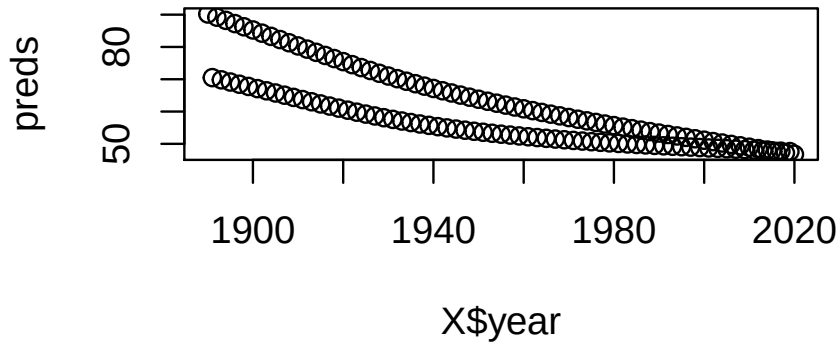
```
plot(time ~ year, data=SwimRecords)
lines(mod, col="red", lwd=2)
lines(mod2, col="blue", lwd=3)
```



Smoothers in k dimensions

GAMS

```
library(gam)
mod <- gam(time ~ sex * s(year, 2), data=SwimRecords)
preds <- predict(mod, newdata=X)
plot(X$year, preds)
```



Programming Activity

Finish up Day 16 Programming Activity.

Look at the smoothness.

Calculate the second derivative and integrate its absolute value over the interval.