Prediction Over Multiple Samples

Contents

5.2 Predictions Over Multiple Samples

We begin by loading some functions that will be used througout this section of the notes.

- `ave_y_mu_sq` calculates the average prediction squared error (APSE) for a single `sample` of data and a predictor function `predfun`

```
ave_y_mu_sq <- function(sample, predfun, na.rm = TRUE){
  mean((sample$y - predfun(sample$x))^2, na.rm = na.rm)
}
```

- `ave_mu_mu_sq` calculates the average squared distance between two predictor functions `predfun1` and `predfun2` across all values in `x`

```
ave_mu_mu_sq <- function(predfun1, predfun2, x, na.rm = TRUE){
  mean((predfun1(x) - predfun2(x))^2, na.rm = na.rm)
}
```

- `getSampleComp` is a function that given a population (`pop`) of size N and a sample `size` n will return an N element vector containing n `TRUE` and $N - n$ `FALSE` values
  - `TRUE` indicates the inclusion of a unit in the sample
  - `FALSE` indicates the inclusion of a unit in the test set

```
getSampleComp <- function(pop, size, replace=FALSE) {
  N <- popSize(pop)
  samp <- rep(FALSE, N)
```

```
  samp[sample(1:N, size, replace = replace)] <- TRUE
  samp
}
```

- `getXYSample` is a function that extracts a given sample of units from a given population and returns a data frame containing just the explanatory variate (x) and the response variate (y)

```
getXYSample <- function(xvarname, yvarname, samp, pop) {
  sampData <- pop[samp, c(xvarname, yvarname)]
  names(sampData) <- c("x", "y")
  sampData
}
```

- `popSize` and `sampSize` are functions used by both functions above to calculate the population and sample sizes, respectively.

```
popSize <- function(pop) {nrow(as.data.frame(pop))}
sampSize <- function(samp) {popSize(samp)}
```

- `getmuhat` is a function that uses least squares to fit polynomial response models (for arbitrary degrees in `x`). Note that this is the version that performs extrapolation and is what was referred to in the previous section as `getmuhat.ext`.

```
getmuhat <- function(sampleXY, complexity = 1) {
  formula <- paste0("y ~ ",
                    if (complexity==0) {
                      "1"
                      } else
                      paste0("poly(x, ", complexity, ", raw = FALSE)")
                      #paste0("bs(x, ", complexity, ")")
  )

  fit <- lm(as.formula(formula), data = sampleXY)
  tx = sampleXY$x
  ty = fit$fitted.values

  range.X = range(tx)
  val.rY  = c( mean(ty[tx == range.X[1]]),
               mean(ty[tx == range.X[2]]) )

  ## From this we construct the predictor function
  muhat <- function(x){
    if ("x" %in% names(x)) {
      ## x is a dataframe containing the variate named
      ## by xvarname
      newdata <- x
    } else
      ## x is a vector of values that needs to be a data.frame
    { newdata <- data.frame(x = x) }
    ## The prediction
    ##
```

```
    val = predict(fit, newdata = newdata)
    val[newdata$x < range.X[1]] = val.rY[1]
    val[newdata$x > range.X[2]] = val.rY[2]
    val
  }
  ## muhat is the function that we need to calculate values
  ## at any x, so we return this function from getmuhat
  muhat
}
```

- `gettauFun` here is a renaming of the `getmuFun` we have previously used. The rationale for renaming this function will be made clear below.

```
gettauFun <- function(pop, xvarname, yvarname){
  pop   = na.omit(pop[, c(xvarname, yvarname)])

  # rule = 2 means return the nearest y-value when extrapolating, same as above.
  # ties = mean means that repeated x-values have their y-values averaged, as above.
  tauFun = approxfun(pop[,xvarname], pop[,yvarname], rule = 2, ties = mean)
  return(tauFun)
}
```

5.2.1 Calculating APSE Over Many Samples

- The inaccuracy of a predictor function $\mu(\mathbf{x})$ is measured by its average prediction squared error:

$$APSE(\mathcal{P}, \widehat{\mu}_{\mathcal{S}}) = \frac{1}{N} \sum_{u \in \mathcal{P}} (y_u - \widehat{\mu}_{\mathcal{S}}(\mathbf{x}_u))^2$$

  - where $\mathcal{S}$ is the sample we used to fit the function, and
  - $\mathcal{T} = \mathcal{P} \setminus \mathcal{S}$ is the complement set of $\mathcal{S}$ such that the population $\mathcal{P} = \mathcal{S} \cup \mathcal{T}$ with $\mathcal{S} \cap \mathcal{T} = \emptyset$

- The function $\widehat{\mu}_{\mathcal{S}}(\mathbf{x})$ is an estimate of $\mu(\mathbf{x})$ based on the single sample $\mathcal{S}$ and
  - its performance depends highly on the particular choice of sample
  - its performance could vary quite a lot from one sample to another

It is important to choose a predictor function that performs well no matter which sample was used to estimate it.

- Suppose that we have many (perhaps all possible) samples $\mathcal{S}_j$ for $j = 1, \ldots, N_{\mathcal{S}}$.
  - For each sample, we can calculate $\widehat{\mu}_{\mathcal{S}_j}(\mathbf{x})$ and hence

$$APSE(\mathcal{P}, \widehat{\mu}_{\mathcal{S}_j})$$

- The average $APSE$ over all $N_{\mathcal{S}}$ samples should be a better measure of the quality of a predictor function.

$$\begin{aligned} APSE(\mathcal{P}, \widetilde{\mu}) &= \frac{1}{N_{\mathcal{S}}} \sum_{j=1}^{N_{\mathcal{S}}} APSE(\mathcal{P}, \widehat{\mu}_{\mathcal{S}_j}) \\ &= \frac{1}{N_{\mathcal{S}}} \sum_{j=1}^{N_{\mathcal{S}}} \frac{1}{N} \sum_{u \in \mathcal{P}} (y_u - \widehat{\mu}_{\mathcal{S}_j}(\mathbf{x}_u))^2 \end{aligned}$$

Note that in $APSE(\mathcal{P}, \widetilde{\mu})$, the **estimator** notation $\widetilde{\mu}$ is used to emphasize that the function is looking at the values of $\widehat{\mu}$ over many (perhaps all possible) samples $\mathcal{S}_j$.

5.2.2 Decomposing $APSE(\mathcal{P}, \widetilde{\mu})$

- In this section we show that the average predicted squared error, $APSE$, for the estimator $\widetilde{\mu}(\mathbf{x})$ is composed of three separate and interpretable pieces.

5.2.2.1 Preamble

- Until now we have thought of prediction in the context of the **response model**

$$y = \mu(\mathbf{x}) + \text{error}$$

where y and $\mathbf{x} = (x_1, \ldots, x_p)$ respectively represent our *observed* response and explanatory variate values

- The (user-specified) predictor function $\mu(\mathbf{x})$ is meant to approximate the **true underlying relationship** between y and $\mathbf{x}$:

$$y = \tau(\mathbf{x})$$

  - the error term in the response model accounts for *model misspecification*, i.e., the difference between $\mu(\mathbf{x})$ and $\tau(\mathbf{x})$

- Note that when we observe the whole population $\mathcal{P}$, there is no uncertainty in the relationship between y and $\mathbf{x}$
  - That is, we observe $\tau(\mathbf{x})$

However, there may still be some uncertainty if, for example, the population contains duplicate $\mathbf{x}$ values that produce different y values

- For this reason, we define $\tau(\mathbf{x})$ to be the conditional average of y given $\mathbf{x}$
  - Suppose that there are K different values of $\mathbf{x}$ in the population $\mathcal{P}$: $\mathbf{x}_1, \ldots, \mathbf{x}_K$
  - Thus the population $\mathcal{P}$ can be partitioned according to the different values of $\mathbf{x}$ as

$$\mathcal{P} = \bigcup_{k=1}^{K} \mathcal{A}_k$$

where

$$\mathcal{A}_k = \{u \ : \ u \in \mathcal{P}, \ \mathbf{x}_u = \mathbf{x}_k\}$$

is the collection of units who all have $\mathbf{x} = \mathbf{x}_k$, for the $k = 1, \ldots, K$

– The conditional average $\tau(\mathbf{x})$ can thus be expressed for each distinct $\mathbf{x}_k$ as

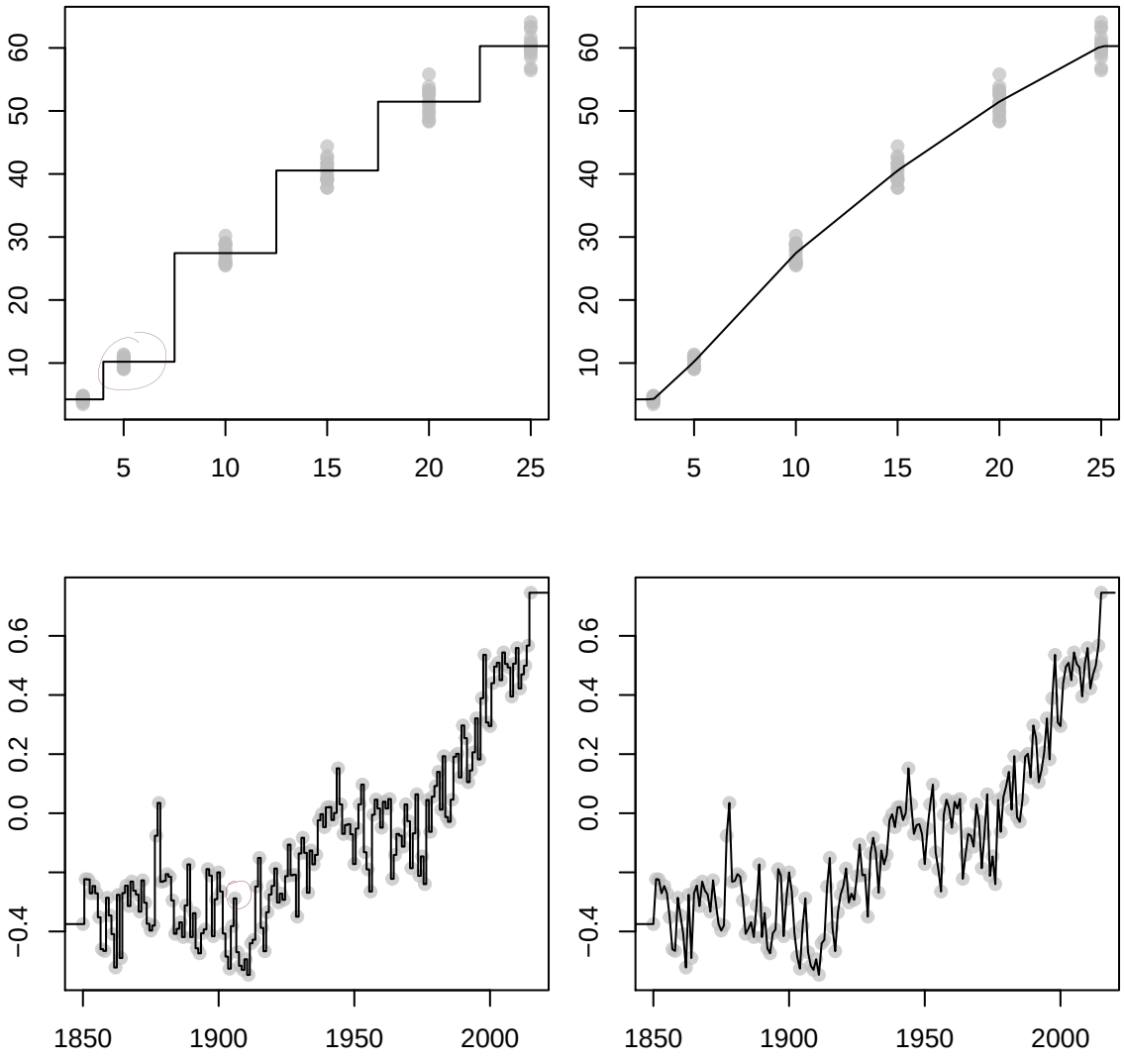
$$\tau(\mathbf{x}_k) = \frac{1}{n_k} \sum_{u \in \mathcal{A}_k} y_u$$

where n_k is the number of units in $\mathcal{A}_k$

- The conditional average $\tau(\mathbf{x})$ defined in this way is in fact the piecewise constant function we've already talked about

– Recall the Loblolly Pine and Global Temperature data:

These plots are of $\tau(x)$



5.2.2.2 The Decomposition

- As mentioned above, $APSE(\mathcal{P}, \widetilde{\mu})$ can be decomposed into three meaningful terms.

- In the derivation that follows it will be useful to note that

$$\overline{\widehat{\mu}}(\mathbf{x}) = \frac{1}{N_{\mathcal{S}}} \sum_{j=1}^{N_{\mathcal{S}}} \widehat{\mu}_{\mathcal{S}_j}(\mathbf{x})$$

  is defined to be the average of the estimated predictor function over all samples

- Here is the decomposition:

$$\begin{aligned}
&APSE(\mathcal{P}, \widetilde{\mu}) \\
&= \tfrac{1}{N_{\mathcal{S}}} \textstyle\sum_{j=1}^{N_{\mathcal{S}}} \tfrac{1}{N} \sum_{u \in \mathcal{P}} (y_u - \widehat{\mu}_{\mathcal{S}_j}(\mathbf{x}_u))^2 \\
&= \tfrac{1}{N_{\mathcal{S}}} \textstyle\sum_{j=1}^{N_{\mathcal{S}}} \tfrac{1}{N} \sum_{u \in \mathcal{P}} (y_u - \tau(\mathbf{x}_u))^2 &&+ \tfrac{1}{N_{\mathcal{S}}} \textstyle\sum_{j=1}^{N_{\mathcal{S}}} \tfrac{1}{N} \sum_{u \in \mathcal{P}} (\widehat{\mu}_{\mathcal{S}_j}(\mathbf{x}_u) - \tau(\mathbf{x}_u))^2 \\
&= \tfrac{1}{N_{\mathcal{S}}} \textstyle\sum_{j=1}^{N_{\mathcal{S}}} \tfrac{1}{N} \sum_{u \in \mathcal{P}} (y_u - \tau(\mathbf{x}_u))^2 &&+ \tfrac{1}{N_{\mathcal{S}}} \textstyle\sum_{j=1}^{N_{\mathcal{S}}} \tfrac{1}{N} \sum_{u \in \mathcal{P}} (\widehat{\mu}_{\mathcal{S}_j}(\mathbf{x}_u) - \overline{\widehat{\mu}}(\mathbf{x}_u))^2 &&+ \tfrac{1}{N_{\mathcal{S}}} \textstyle\sum_{j=1}^{N_{\mathcal{S}}} \tfrac{1}{N} \sum_{u \in \mathcal{P}} (\overline{\widehat{\mu}}(\mathbf{x}_u) - \tau(\mathbf{x}_u))^2 \\
&= \tfrac{1}{N} \textstyle\sum_{u \in \mathcal{P}} (y_u - \tau(\mathbf{x}_u))^2 &&+ \tfrac{1}{N_{\mathcal{S}}} \textstyle\sum_{j=1}^{N_{\mathcal{S}}} \tfrac{1}{N} \sum_{u \in \mathcal{P}} (\widehat{\mu}_{\mathcal{S}_j}(\mathbf{x}_u) - \overline{\widehat{\mu}}(\mathbf{x}_u))^2 &&+ \tfrac{1}{N} \textstyle\sum_{u \in \mathcal{P}} (\overline{\widehat{\mu}}(\mathbf{x}_u) - \tau(\mathbf{x}_u))^2
\end{aligned}$$

- The second term is clearly reflects the variability of the estimator $\widetilde{\mu}$ and the third term clearly reflects the estimator's bias (squared), but to interpret the first term, let us consider rewriting the sum over $u \in \mathcal{P}$ as a sum of u over the unique $\mathbf{x}$ values:

$$\frac{1}{N} \sum_{u \in \mathcal{P}} (y_u - \tau(\mathbf{x}_u))^2 = \sum_{k=1}^{K} \frac{n_k}{N} \sum_{u \in \mathcal{A}_k} \frac{1}{n_k} (y_u - \tau(\mathbf{x}_k))^2$$

  which we can interpret as the conditional variance of y given $\mathbf{x}$, averaged over all of the unique $\mathbf{x}$ values

- Thus we find that

$$\begin{aligned}
&APSE(\mathcal{P}, \widetilde{\mu}) \\
&= \textstyle\sum_{k=1}^{K} \tfrac{n_k}{N} \sum_{u \in \mathcal{A}_k} \tfrac{1}{n_k} (y_u - \tau(\mathbf{x}_k))^2 &&+ \tfrac{1}{N_{\mathcal{S}}} \textstyle\sum_{j=1}^{N_{\mathcal{S}}} \tfrac{1}{N} \sum_{u \in \mathcal{P}} (\widehat{\mu}_{\mathcal{S}_j}(\mathbf{x}_u) - \overline{\widehat{\mu}}(\mathbf{x}_u))^2 &&+ \tfrac{1}{N} \textstyle\sum_{u \in \mathcal{P}} (\overline{\widehat{\mu}}(\mathbf{x}_u) - \tau(\mathbf{x}_u))^2 \\
&\equiv Ave_{\mathbf{x}} \left(Var[Y|\mathbf{x}] \right) &&+ \quad Var[\widetilde{\mu}] &&+ Bias^2[\widetilde{\mu}]
\end{aligned}$$

  where

  - $Ave_{\mathbf{x}} \left(Var[Y|\mathbf{x}] \right)$ is the average of the conditional variance of the response y
  - $Var[\widetilde{\mu}]$ is the variance of the estimator
  - $Bias^2[\widetilde{\mu}]$ is the squared bias of the estimator

Note: The first term, $Ave_{\mathbf{x}} \left(Var[Y|\mathbf{x}] \right)$, is independent of $\mu(\mathbf{x})$ and thus will not change for different specifications of the predictor function.

5.2.2.3 Taking the Decomposition One Step Further

- In Section 5.1.3 we showed that $APSE(\mathcal{P}, \widehat{\mu}_{\mathcal{P}})$ could be decomposed into two components based on $\mathcal{S}$ and $\mathcal{T}$ since $\mathcal{P} = \mathcal{S} \cup \mathcal{T}$
  - We can do that here too:

$$
\begin{aligned}
&APSE(\mathcal{P}, \widetilde{\mu}) \\
&= \frac{1}{N_{\mathcal{S}}} \sum_{j=1}^{N_{\mathcal{S}}} \frac{1}{N} \sum_{u \in \mathcal{P}} (y_u - \tau(\mathbf{x}_u))^2 + \frac{1}{N_{\mathcal{S}}} \sum_{j=1}^{N_{\mathcal{S}}} \frac{1}{N} \sum_{u \in \mathcal{P}} (\widehat{\mu}_{\mathcal{S}_j}(\mathbf{x}_u) - \overline{\widehat{\mu}}(\mathbf{x}_u))^2 + \frac{1}{N_{\mathcal{S}}} \sum_{j=1}^{N_{\mathcal{S}}} \frac{1}{N} \sum_{u \in \mathcal{P}} (\overline{\widehat{\mu}}(\mathbf{x}_u) - \tau(\mathbf{x}_u))^2 \\
&= \frac{n}{N} \left\{ \frac{1}{N_{\mathcal{S}}} \sum_{j=1}^{N_{\mathcal{S}}} \frac{1}{n} \sum_{u \in \mathcal{S}} (y_u - \tau(\mathbf{x}_u))^2 + \frac{1}{N_{\mathcal{S}}} \sum_{j=1}^{N_{\mathcal{S}}} \frac{1}{n} \sum_{u \in \mathcal{S}} (\widehat{\mu}_{\mathcal{S}_j}(\mathbf{x}_u) - \overline{\widehat{\mu}}(\mathbf{x}_u))^2 + \frac{1}{N_{\mathcal{S}}} \sum_{j=1}^{N_{\mathcal{S}}} \frac{1}{n} \sum_{u \in \mathcal{S}} (\overline{\widehat{\mu}}(\mathbf{x}_u) - \tau(\mathbf{x}_u))^2 \right\} \\
&+ \left(\frac{N-n}{N}\right) \left\{ \frac{1}{N_{\mathcal{S}}} \sum_{j=1}^{N_{\mathcal{S}}} \frac{1}{N-n} \sum_{u \in \mathcal{T}} (y_u - \tau(\mathbf{x}_u))^2 + \frac{1}{N_{\mathcal{S}}} \sum_{j=1}^{N_{\mathcal{S}}} \frac{1}{N-n} \sum_{u \in \mathcal{T}} (\widehat{\mu}_{\mathcal{S}_j}(\mathbf{x}_u) - \overline{\widehat{\mu}}(\mathbf{x}_u))^2 + \frac{1}{N_{\mathcal{S}}} \sum_{j=1}^{N_{\mathcal{S}}} \frac{1}{N-n} \sum_{u \in \mathcal{T}} (\overline{\widehat{\mu}}(\mathbf{x}_u) -
\end{aligned}
$$

- We might write

$$
\begin{aligned}
APSE(\mathcal{P}, \widetilde{\mu}) = &\left(\frac{n}{N}\right) \left\{ \widehat{APSE}(\mathcal{P}, \widetilde{\mu}) \quad \text{based on the } \textbf{same} \text{ samples used by } \widehat{\mu} \right\} \\
&+ \left(\frac{N-n}{N}\right) \left\{ \widehat{APSE}(\mathcal{P}, \widetilde{\mu}) \quad \text{based on samples } \textbf{not} \text{ used by } \widehat{\mu} \right\}
\end{aligned}
$$

- Notice that if $n << N$, then the second term dominates.
  - But regardless of the size of n we may sometimes want to focus our evaluation only on the second term since this evaluation is based only on values not used in the actual estimation process.
  - This provides the most fair assessment of the model's **out of sample** performance.

5.2.3 Understanding Each of These Pieces

5.2.3.1 Intuition: $\tau(\mathbf{x})$

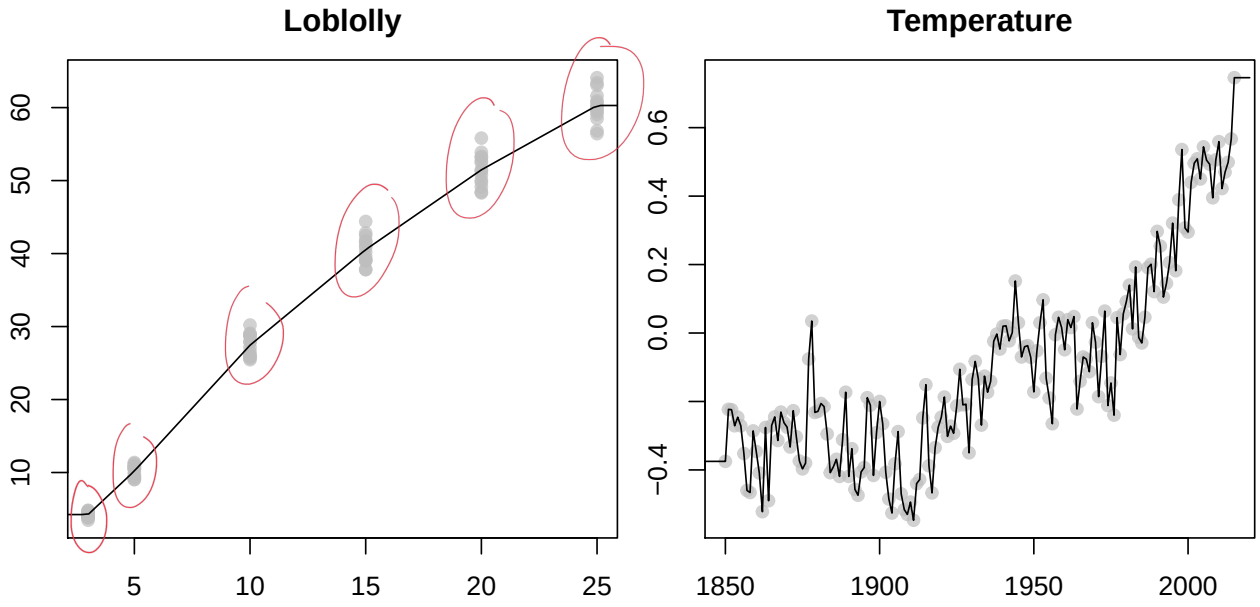
- Recall the Loblolly Pine and Global Temperature data
  - One qualitative difference between these two datasets is that the Loblolly data has duplicate $\mathbf{x}$ values, whereas all of the $\mathbf{x}$ values in the Temperature data are unique
  - And in particular, for the Loblolly data we have multiple different y values for any given $\mathbf{x}$.
  - This facilitates computation of $Ave_{\mathbf{x}}\left(Var[Y|\mathbf{x}]\right)$.

- The conditional average

$$
\tau(\mathbf{x}_k) = \frac{1}{n_k} \sum_{u \in \mathcal{A}_k} y_u \quad \text{where} \quad \mathcal{A}_k = \{u \ : \ u \in \mathcal{P}, \ \mathbf{x}_u = \mathbf{x}_k\}
$$

  is visualized for each of these datasets below:

```
par(mfrow=c(1,2), mar=2.5*c(1,1,1,0.1))

plot(Loblolly$age[order(Loblolly$age)], Loblolly$height[order(Loblolly$age)], xlab="Age",
     ylab="Height", col=adjustcolor("grey", 0.7),
     pch=19, main="Loblolly")
tau.age = gettauFun(pop=Loblolly, xvarname="age", yvarname="height")
curve(tau.age, 0, 30, add=TRUE, n=100)

plot(temperature$YEAR, temperature$ANNUAL, xlab="Year", ylab="Annual Temperature",
     col=adjustcolor("grey", 0.7), pch=19, main="Temperature")
tau.annual = gettauFun(pop=temperature, xvarname="YEAR", yvarname="ANNUAL")
curve(tau.annual, 1840, 2020, add=TRUE, 181)
```



- $Ave_x(Var[Y|x]) = \frac{1}{N}\sum_{u\in\mathcal{P}}(y_u - \tau(x_u))^2 = \frac{1}{N}\sum_{k=1}^{K}\sum_{u\in\mathcal{A}_k}(y_u - \tau(\mathbf{x}_k))^2$ is

```
mat = matrix(0, nrow=2,ncol=1)
mat[,1] = c(mean( (Loblolly$height - tau.age(Loblolly$age))^2  ),
mean( (temperature$ANNUAL - tau.annual(temperature$YEAR))^2  ) )
rownames(mat) = c("Loblolly", "Temperature")
colnames(mat) = "Ave_x Var(Y|x)"
round(mat,2)
```

```
##             Ave_x Var(Y|x)
## Loblolly                2.64
## Temperature             0.00
```

- What does $Var(Y|\mathbf{x}) = 0$ mean?

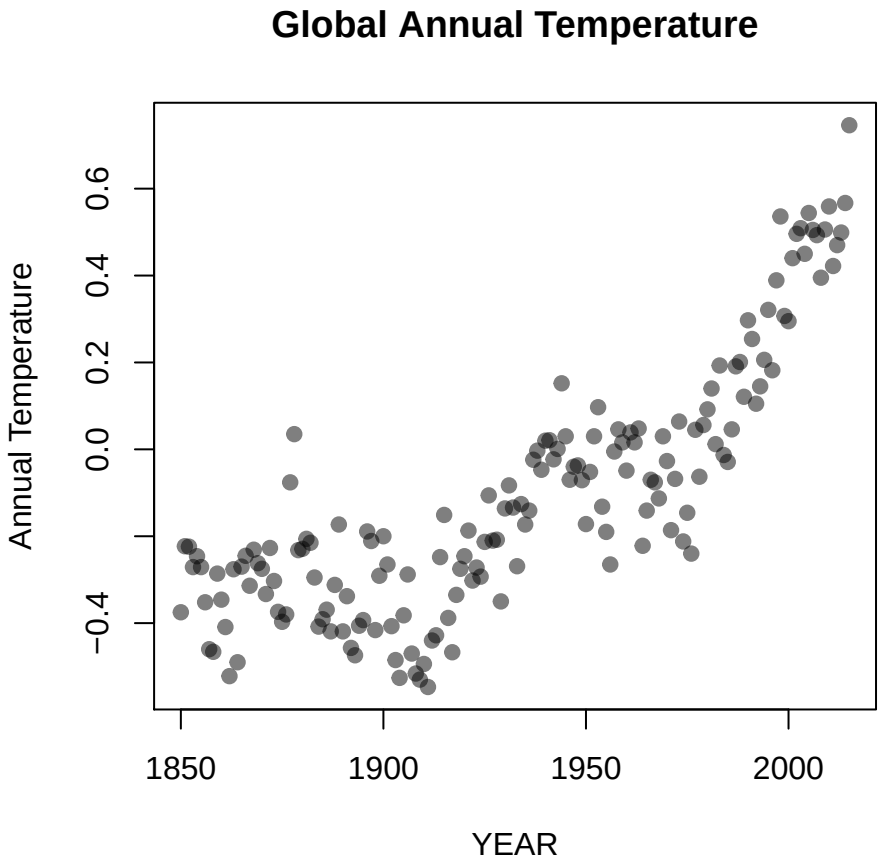
1. All x values are unique (i.e., no duplication)

2. Duplicate y values at unique x values are all the same

5.2.3.2 Intuition: $Var[\widetilde{\mu}]$

- Consider the Global Temperature data.

```
plot(temperature$YEAR, temperature$ANNUAL, xlab="YEAR", ylab="Annual Temperature",
     main="Global Annual Temperature", col=adjustcolor("black", 0.5), pch=19 )
```



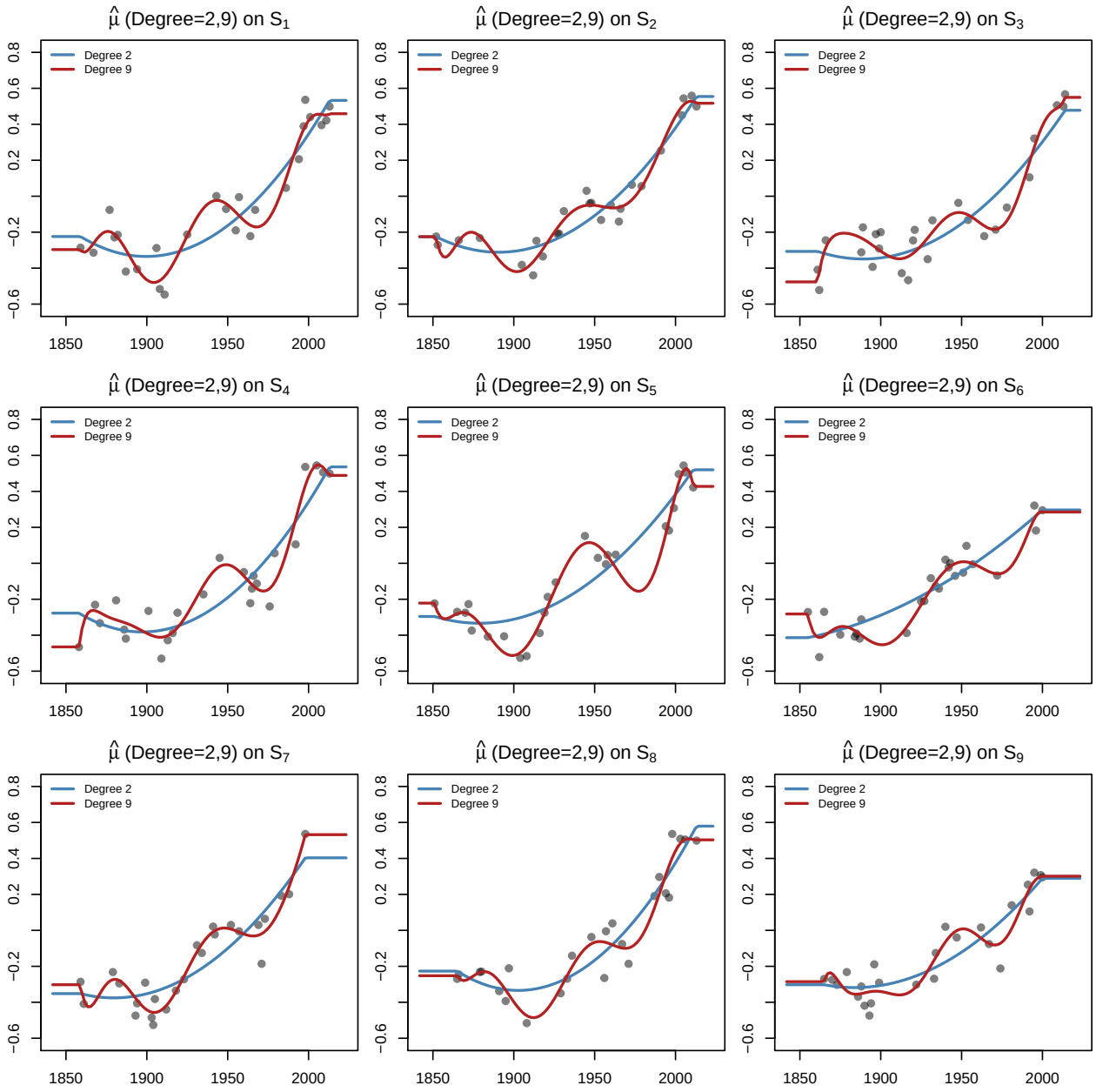
- Let us take $N_{\mathcal{S}} = 9$ samples of size $n = 25$ from this population

```
xnam <- "YEAR"
ynam <- "ANNUAL"
pop  <- temperature
n    <- 25
N_S  <- 9

set.seed(1)  # for reproducibility
samples <- lapply(1:N_S, FUN= function(i){getSampleComp(pop, n)})
Ssam    <- lapply(samples, FUN= function(Si){getXYSample(xnam, ynam, Si, pop)})
Tsam    <- lapply(samples, FUN= function(Si){getXYSample(xnam, ynam, !Si, pop)})
```

- And then fit polynomials of degree 2 and 9 to each of these samples

```
muhats2 <- lapply(Ssam, getmuhat, complexity = 2)
muhats9 <- lapply(Ssam, getmuhat, complexity = 9)

par(mfrow=c(3,3), mar=2.5*c(1,1,1,0.1))
```

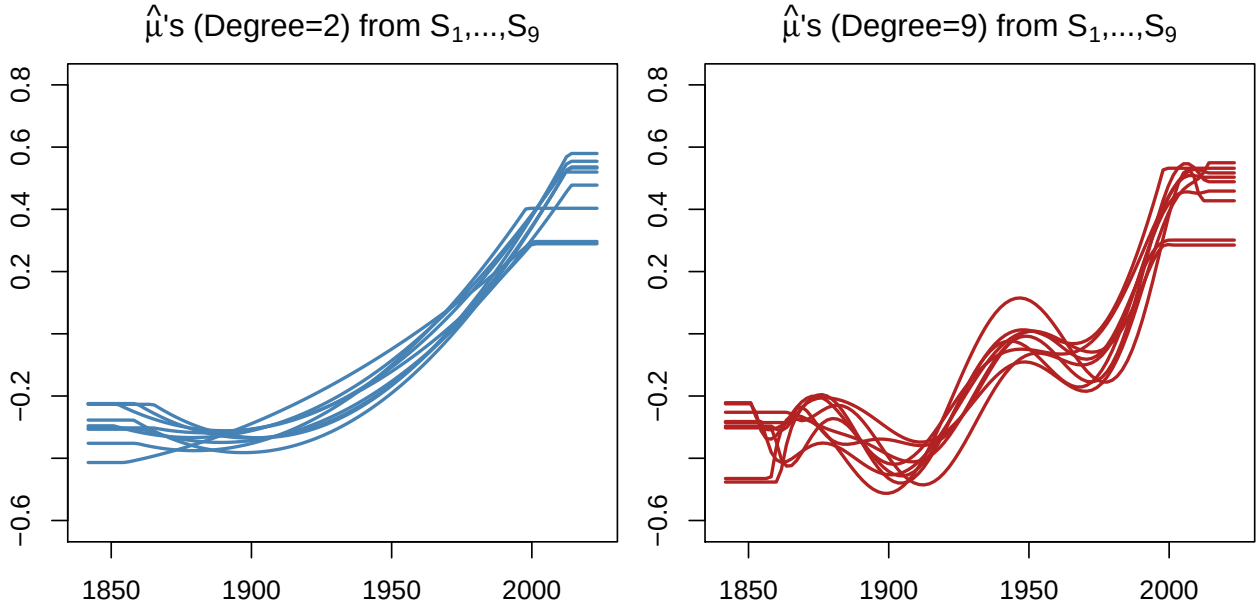
```r
xlim <- extendrange(temperature[, xnam])
ylim <- extendrange(temperature[, ynam])

for (i in 1:N_S) {
    plot(Ssam[[i]],
     main = bquote(hat(mu) ~ "(Degree=2,9) on S"[.(i)]),
     xlab = xnam, ylab = ynam,
     pch=19, col= adjustcolor("black", 0.5), ylim=ylim, xlim=xlim)
   tempfn =  muhats2[[i]]
   curve(tempfn, from = xlim[1], to = xlim[2], add = TRUE, col="steelblue", lwd=2)
   tempfn =  muhats9[[i]]
   curve(tempfn, from = xlim[1], to = xlim[2], add = TRUE, col="firebrick", lwd=2)
   legend("topleft", col = c("steelblue", "firebrick"), lwd = 2,
          legend = c("Degree 2", "Degree 9"), bty = "n", cex = 0.8)
}
```



- To better visualize the **sampling variability** let us plot each of the $N_{\mathcal{S}} = 9$ curves (for a given degree) on the same plot.

```
par(mfrow=c(1,2), mar=2.5*c(1,1,1,0.1))

plot(Ssam[[i]], main = bquote(hat(mu)*"'s (Degree=2) from S"[1]*",...,S"[9]), xlab = xnam,
     ylab = ynam, pch=19, col=0, ylim=ylim, xlim=xlim)

for (i in 1:N_S) {
   tempfn =  muhats2[[i]]
   curve(tempfn, from = xlim[1], to = xlim[2], add = TRUE, col="steelblue", lwd=2)
}
```

```
plot(Ssam[[i]], main = bquote(hat(mu)*"'s (Degree=9) from S"[1]*",...,S"[9]), xlab = xnam,
     ylab = ynam, pch=19, col=0, ylim=ylim, xlim=xlim)
for (i in 1:N_S) {
   tempfn =  muhats9[[i]]
   curve(tempfn, from = xlim[1], to = xlim[2], add = TRUE, col="firebrick", lwd=2)
}
```



- Let's extend this example and consider more than just `degree=2` and `degree=9`; let's consider `degree=1:9`

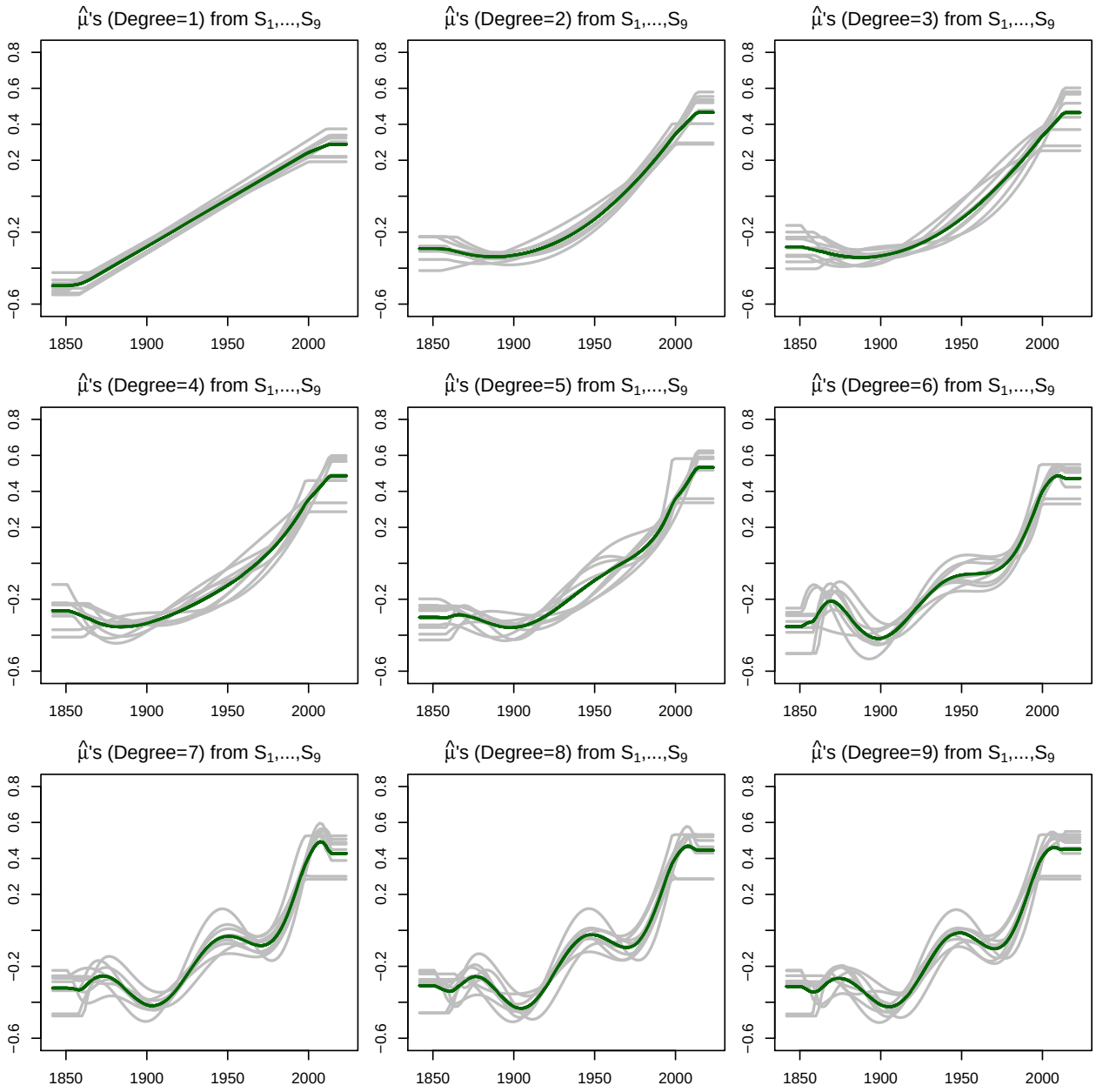
```
degrees <- 1:9
muhats <- Map( function(i) {lapply(Ssam, getmuhat, complexity = i)}, degrees)
```

- For each `degree` let's also calculate the *average* fitted polynomial

$$\overline{\widehat{\mu}}(x) = \frac{1}{N_{\mathcal{S}}} \sum_{j=1}^{N_{\mathcal{S}}} \widehat{\mu}_{\mathcal{S}_j}(x)$$

  - Grey lines below are $\widehat{\mu}_{\mathcal{S}_j}(x)$ for $j = 1, \ldots, N_{\mathcal{S}}$
  - Green lines below is $\overline{\widehat{\mu}}(x)$

```
getmubar <- function(muhats) {
  function(x) {
    Ans <- sapply(muhats, FUN=function(muhat){muhat(x)})
    apply(Ans, MARGIN=1, FUN=mean)
  }
}
```



- What do you observe in these plots?

  Sampling variation Exists!
  Up The grey lines are not all the same

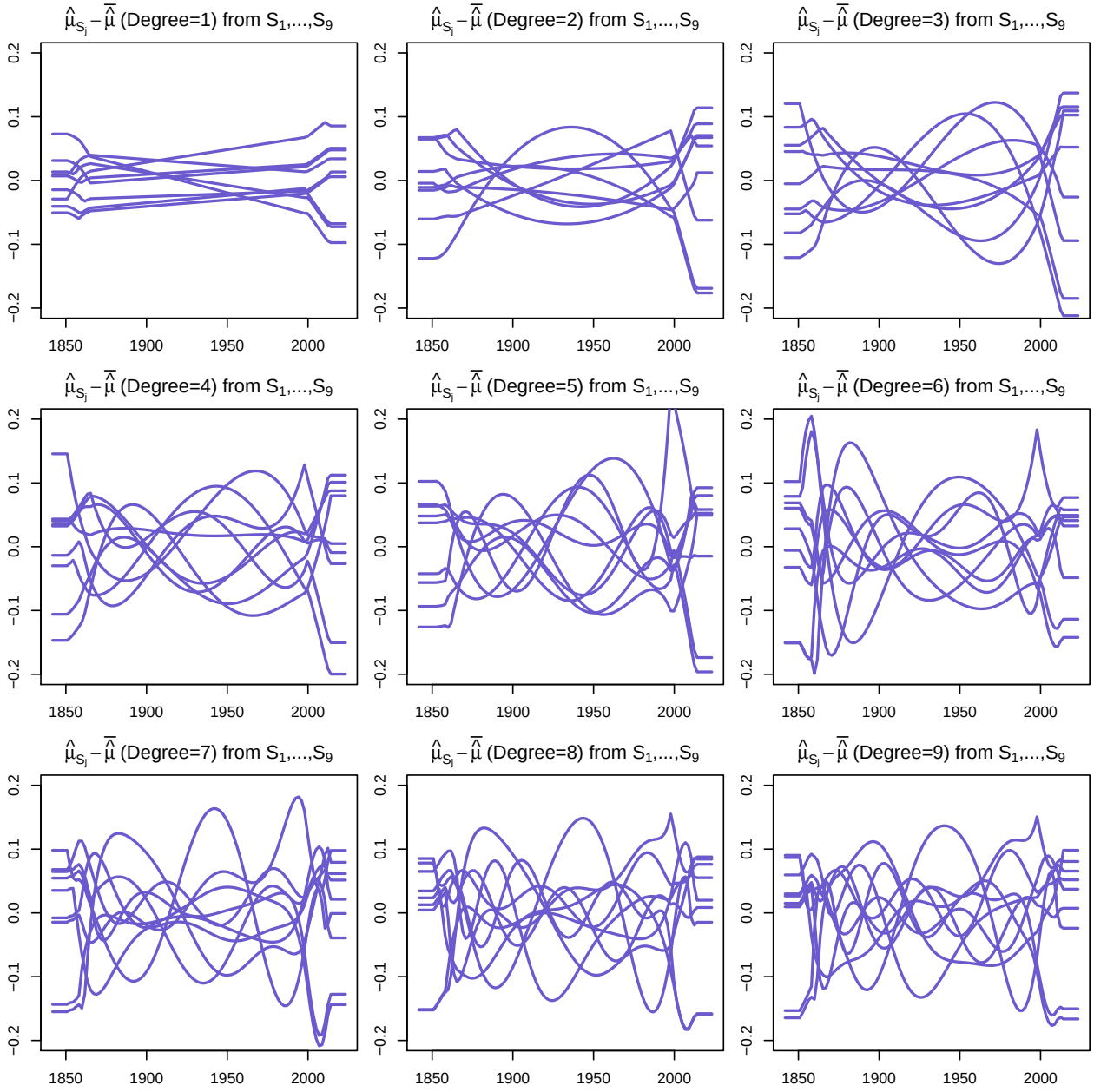
- We may also visualize the sampling variability by examining the differences

$$\widehat{\mu}_{\mathcal{S}_j}(x) - \overline{\widehat{\mu}}(x)$$

  for each polynomial and each sample $\mathcal{S}_j$, $j = 1, \ldots, N_{\mathcal{S}}$.

  – These are differences between a grey line and the corresponding green line

```
par(mfrow=c(3,3), mar=2.5*c(1,1,1,0.1))

for (i in degrees) {
  plot(Ssam[[i]], main=bquote(hat(mu)["S"[j]] - bar(hat(mu)) ~ "(Degree="*.(i)*") from S"[1]*",...,S"[9]
       xlab = xnam, ylab = ynam, pch=19, col=0, ylim=c(-0.2,0.2), xlim=xlim)
    for (j in 1:N_S) {
   tempfn1 =  muhats[[i]][[j]]
   tempfn2 =  mubars[[i]]
   tempnew = function(z) { tempfn1(z) - tempfn2(z)}
   curve(tempnew, from = xlim[1], to = xlim[2], add = TRUE, col="slateblue", lwd=2)
}}
```



- We can numerically calculate $Var[\widetilde{\mu}]$ using the function `var_mutilde` defined below

```
var_mutilde <- function(Ssamples, Tsamples, complexity){
  ## get the predictor function for every sample S
  muhats <- lapply(Ssamples,
                   FUN=function(sample){
                     getmuhat(sample, complexity)
                   }
  )
  ## get the average of these, mubar
  mubar <- getmubar(muhats)

  ## average over all samples S
  N_S <- length(Ssamples)
  mean(sapply(1:N_S,
              FUN=function(j){
                ## get muhat based on sample S_j
                muhat <- muhats[[j]]
                ## average over (x_i,y_i) in a
                ## single sample T_j the squares
                ## (y - muhat(x))^2
                T_j <- Tsamples[[j]]
                ave_mu_mu_sq(muhat, mubar, T_j$x)
              }
  )
  )
}
```

- Applying this function to our example yields the following results:

```
degrees <- 1:9

varmu <-  sapply(degrees,
     FUN = function(deg){ var_mutilde(Ssam, Tsam, complexity = deg)} )

Vmat = matrix(varmu, nrow=1, dimnames=list("Var(mu)", paste("deg=", degrees, sep="") ) )
round(Vmat,4)
```

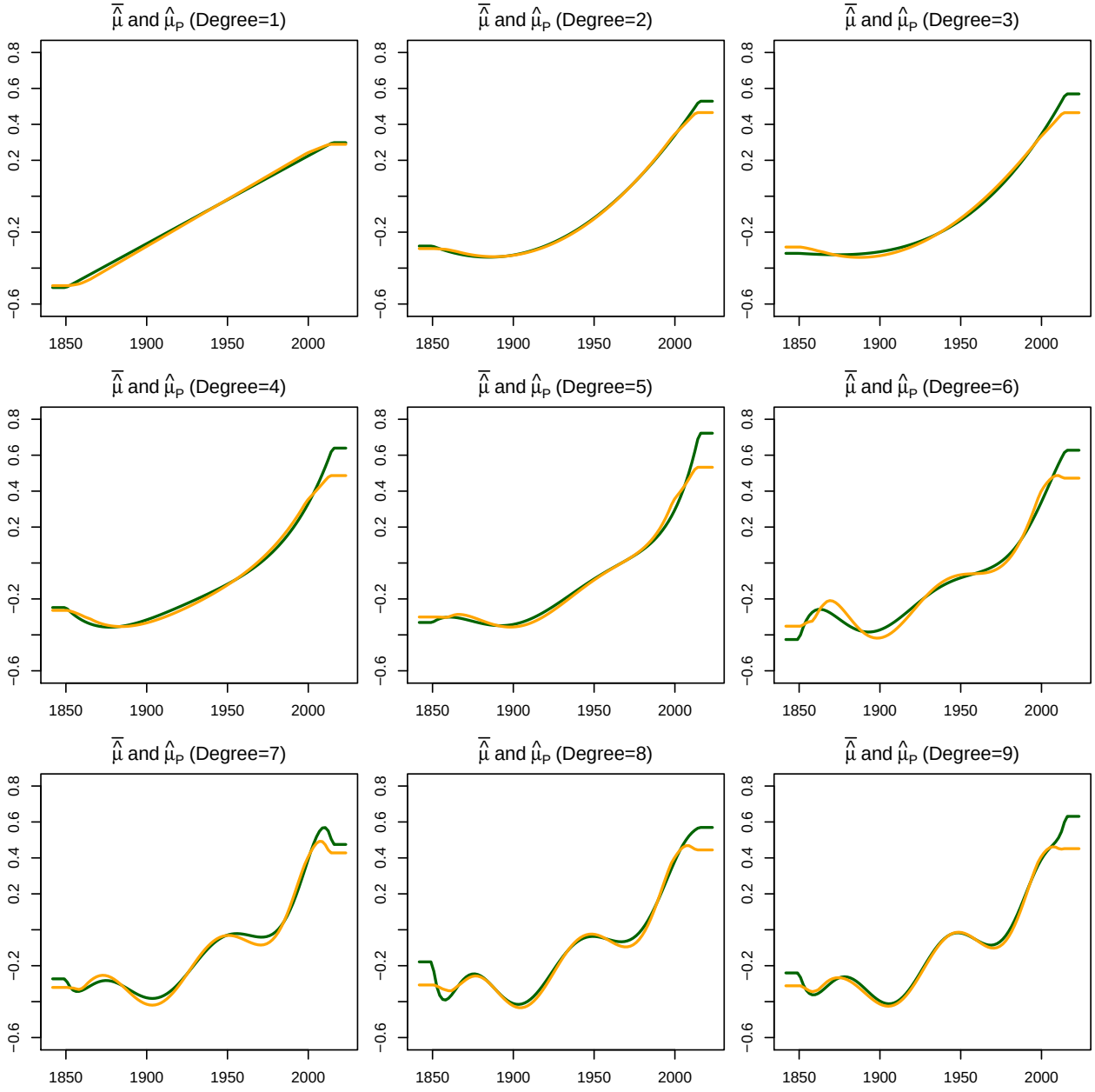
```
##          deg=1 deg=2  deg=3  deg=4  deg=5  deg=6  deg=7  deg=8  deg=9
## Var(mu) 0.001 0.002 0.0034 0.0034 0.0039 0.0045 0.0044 0.0045 0.0042
```

ASIDE:

- The average function $\overline{\widehat{\mu}}(x)$ is different from the function fitted on the whole population $\widehat{\mu}_{\mathcal{P}}(x)$.
- For the same degree, each plot below shows a
  - Green line, representing the average fitted function $\overline{\widehat{\mu}}(x)$
  - Orange line, representing the function to the entire population $\widehat{\mu}_{\mathcal{P}}(x)$

```
par(mfrow=c(3,3), mar=2.5*c(1,1,1,0.1))

for (i in degrees ) {
```

```
    plot(Ssam[[i]],
         main = bquote(bar(hat(mu)) ~ "and" ~ hat(mu)[P] ~ "(Degree="*.(i)*")"),
         xlab = xnam, ylab = ynam, pch=19, col=0, ylim=ylim, xlim=xlim)
    tempfn = getmuhat(getXYSample(xnam, ynam, rep(TRUE,166), pop), complexity = i)
    curve(tempfn, from = xlim[1], to = xlim[2], add = TRUE, col="darkgreen", lwd=2)
    tempfn =  mubars[[i]]
    curve(tempfn, from = xlim[1], to = xlim[2], add = TRUE, col="orange", lwd=2)
}
```



- Note, also, neither of these are the true function $\tau(x)$!
  - $\tau(x)$ is the piecewise function (at least, that is our best guess of it).

5.2.3.3 Intuition: $Bias[\widehat{\mu}]$

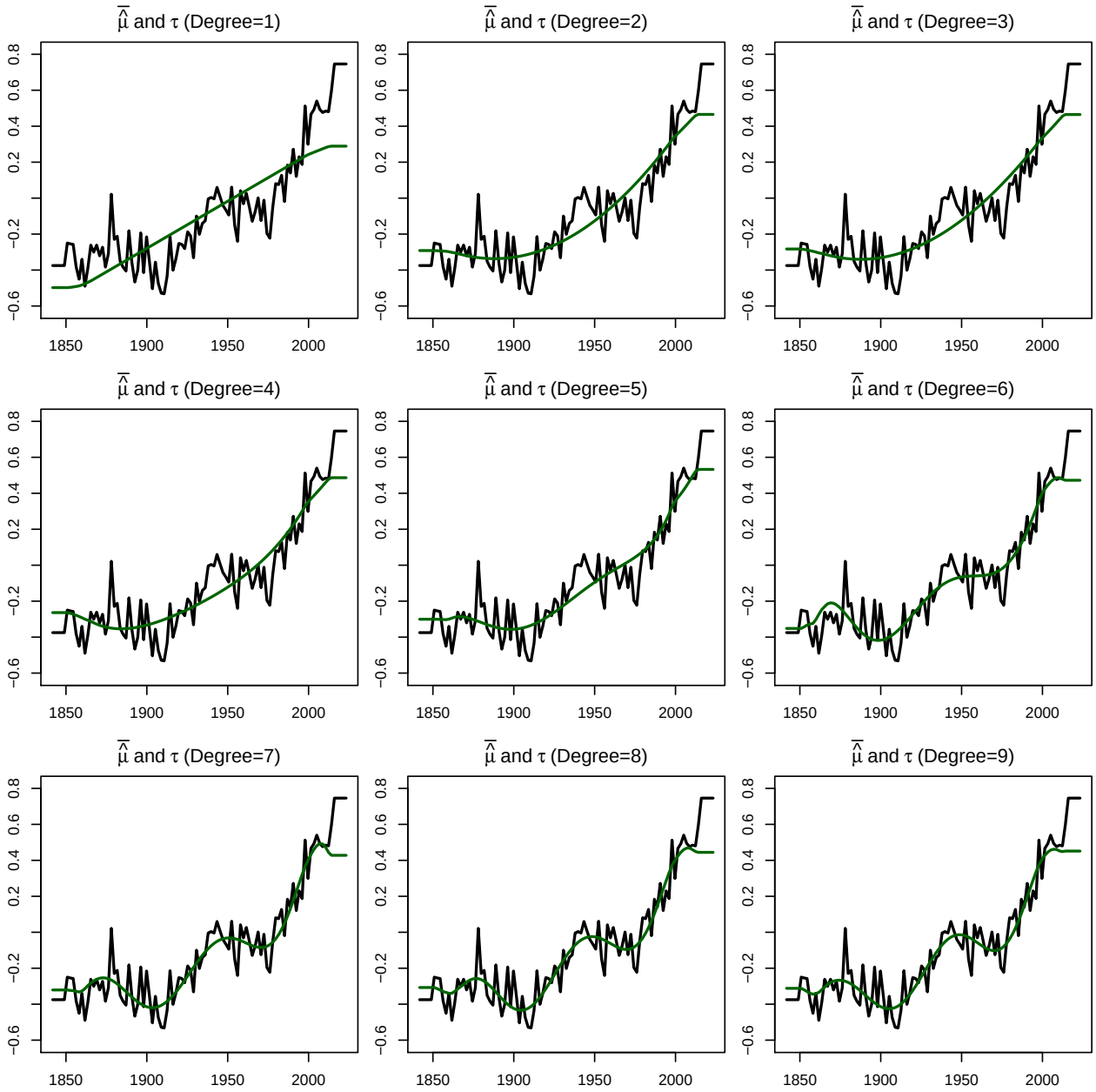
- The bias concerns the difference

$$\overline{\widehat{\mu}}(\mathbf{x}_u) - \tau(\mathbf{x}_u)$$

- For each degree in `1:9` let's visually compare $\overline{\widehat{\mu}}(x)$ to $\tau(\mathbf{x})$

- For the same degree, each plot below shows a
  - Green line representing $\overline{\widehat{\mu}}(x) = \frac{1}{N_S}\sum_{j=1}^{N_S} \widehat{\mu}_{\mathcal{S}_j}(x)$
  - Black line representing $\tau(x)$: the piecewise function defined on $\mathcal{P}$

```
par(mfrow=c(3,3), mar=2.5*c(1,1,1,0.1))

for (i in degrees) {
   plot(Ssam[[i]],
        main = bquote(bar(hat(mu)) ~ "and" ~ tau ~ "(Degree="*.(i)*")"),
        xlab = xnam, ylab = ynam, pch=19, col=0, ylim=ylim, xlim=xlim)
   tempfn = gettauFun(temperature, xnam, ynam)
   curve(tempfn, from = xlim[1], to = xlim[2], add = TRUE, col="black", lwd=2)
   tempfn =  mubars[[i]]
   curve(tempfn, from = xlim[1], to = xlim[2], add = TRUE, col="darkgreen", lwd=2)
}
```

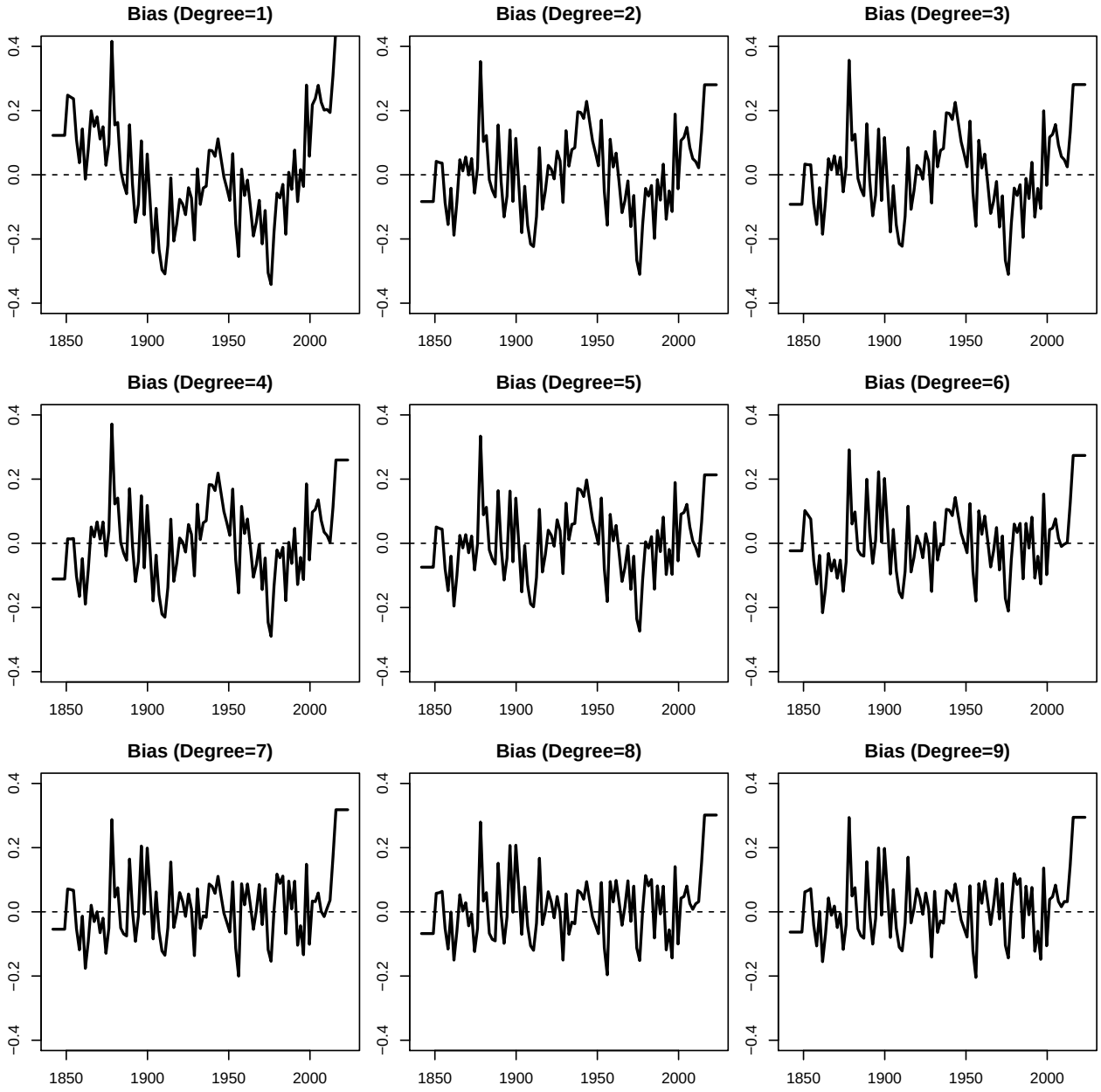


- This bias may be better visualized by plotting the **bias**

$$\overline{\hat{\mu}}(x) - \tau(x)$$

```
par(mfrow=c(3,3), mar=2.5*c(1,1,1,0.1))

for (i in degrees) {
   plot(Ssam[[i]],  main=paste0("Bias (Degree=", i,")"),
        xlab = xnam, ylab = ynam, pch=19, col=0, ylim=c(-0.4, 0.4), xlim=xlim)
   tempfn1 = gettauFun(temperature, xnam, ynam)
   tempfn2 =  mubars[[i]]
   tempnew = function(z) { tempfn1(z) - tempfn2(z)}
   curve(tempnew, from = xlim[1], to = xlim[2], add = TRUE, col="black", lwd=2)
```

```
  abline(h=0, lty=2)
}
```



- We can numerically calculate $Bias[\widetilde{\mu}]$ using the function `bias2_mutilde` defined below

```
bias2_mutilde <- function(Ssamples, Tsamples, tau, complexity) {
  ## get the predictor function for every sample S
  muhats <- lapply(Ssamples,
                   FUN=function(sample) getmuhat(sample, complexity)
  )
  ## get the average of these, mubar
  mubar <- getmubar(muhats)
```

```
  ## average over all samples S
  N_S <- length(Ssamples)
  mean(sapply(1:N_S,
              FUN=function(j){
                ## average over (x_i,y_i) in a
                ## single sample T_j the squares
                ## (y - muhat(x))^2
                T_j <- Tsamples[[j]]
                ave_mu_mu_sq(mubar, tau, T_j$x)
              }
  )
  )
}
```

- Applying this function to our example yields the following results:

```
tau.annual = gettauFun(pop, xnam, ynam)
bias2 <-  sapply(degrees,
     FUN = function(deg){ bias2_mutilde(Ssam, Tsam, complexity = deg, tau = tau.annual)} )

Bmat = matrix(bias2, nrow=1, dimnames=list("[Bias(mu)]^2", paste("deg=", degrees, sep="") ) )
round(Bmat,4)
```

```
##                deg=1  deg=2  deg=3  deg=4  deg=5  deg=6  deg=7  deg=8 deg=9
## [Bias(mu)]^2 0.0265 0.0156 0.0156 0.0148 0.0127 0.0112 0.0102 0.0099  0.01
```

- Which degree polynomial is associated with the smallest bias?

In general as complexity ↗ bias^2 ↘

5.2.3.4 Putting It All Together

- We can combine the three components
  - $Ave_{\mathbf{x}}\left(Var[Y|\mathbf{x}]\right)$
  - $Var[\widetilde{\mu}]$
  - $Bias^2[\widetilde{\mu}]$

  into a single measure to obtain the APSE.

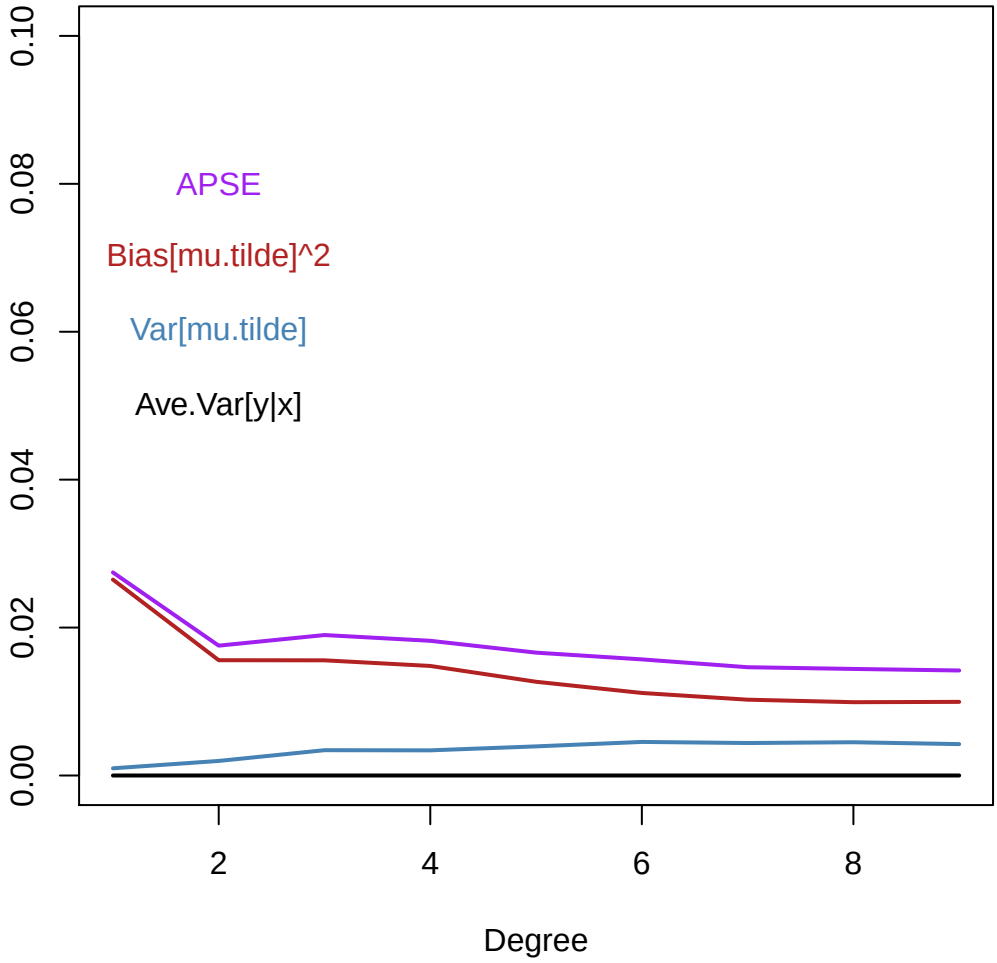
```
BVmat = rbind(0,Bmat, Vmat, Bmat+Vmat)
rownames(BVmat)[1] = "Var(y|x)"
rownames(BVmat)[4] = "APSE"
round(BVmat,3)
```

```
##               deg=1 deg=2 deg=3 deg=4 deg=5 deg=6 deg=7 deg=8 deg=9
## Var(y|x)      0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
## [Bias(mu)]^2  0.026 0.016 0.016 0.015 0.013 0.011 0.010 0.010 0.010
## Var(mu)       0.001 0.002 0.003 0.003 0.004 0.005 0.004 0.004 0.004
## APSE          0.027 0.018 0.019 0.018 0.017 0.016 0.015 0.014 0.014
```

- Recall that $Var(y|x) = 0$. Why?

- Let's visualize these results

```
plot( 1:9, BVmat[2,], xlab="Degree", ylab="", type='l', ylim=c(0, 0.1), col="firebrick", lwd=2 )
lines( 1:9, (BVmat[1,]^2), xlab="Degree", ylab="", col="black", lwd=2 )
lines( 1:9, BVmat[3,], xlab="Degree", ylab="", col="steelblue", lwd=2 )
lines( 1:9, BVmat[4,], col="purple", lwd=2)
text(2,0.08,'APSE',col="purple")
text(2,0.07,'Bias[mu.tilde]^2',col="firebrick")
text(2,0.06,'Var[mu.tilde]',col="steelblue")
text(2,0.05,'Ave.Var[y|x]',col="black")
```



- The results above can be replicated and simplified using the single function `apse_all` which calculates APSE and each of its components all at once:

```
apse_all <- function(Ssamples, Tsamples, complexity, tau){
  ## average over the samples S
  ##
  N_S <- length(Ssamples)
  muhats <- lapply(Ssamples,
                   FUN=function(sample) getmuhat(sample, complexity)
```

```
  )
  ## get the average of these, mubar
  mubar <- getmubar(muhats)

  rowMeans(sapply(1:N_S,
                  FUN=function(j){
                    T_j <- Tsamples[[j]]
                    S_j <- Ssamples[[j]]
                    muhat <- muhats[[j]]
                    ## Take care of any NAs
                    T_j <- na.omit(T_j)
                    y <- c(S_j$y, T_j$y)
                    x <- c(S_j$x, T_j$x)

                    tau_x    <- tau(x)
                    muhat_x <- muhat(x)
                    mubar_x <- mubar(x)

                    apse        <- (y - muhat_x)
                    bias2       <- (mubar_x - tau_x)
                    var_mutilde <-  (muhat_x - mubar_x)
                    var_y       <- (y - tau_x)

                    squares <- rbind(apse, var_mutilde, bias2, var_y)^2

                    ## return means
                    rowMeans(squares)
                  }
  ))
}
```

- Applying this function to our example (and increasing the degrees considered) yields the following results:

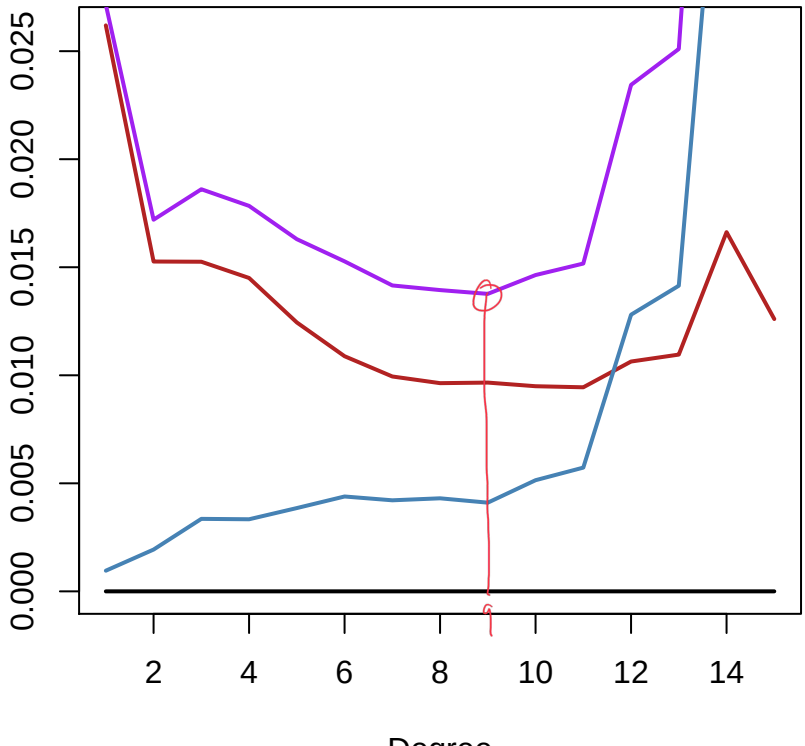
```
degrees <- 1:15
tau.annual = gettauFun(pop, xnam, ynam)

apse_vals <-  sapply(degrees, FUN = function(deg){
              apse_all(Ssam, Tsam, complexity = deg, tau = tau.annual)
    } )

colnames(apse_vals)      =  paste("deg=", degrees, sep="")


round(apse_vals,3)
```

```
##             deg=1 deg=2 deg=3 deg=4 deg=5 deg=6 deg=7 deg=8 deg=9 deg=10
## apse        0.027 0.017 0.019 0.018 0.016 0.015 0.014 0.014 0.014  0.015
## var_mutilde 0.001 0.002 0.003 0.003 0.004 0.004 0.004 0.004 0.004  0.005
## bias2       0.026 0.015 0.015 0.015 0.012 0.011 0.010 0.010 0.010  0.009
## var_y       0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000  0.000
##             deg=11 deg=12 deg=13 deg=14 deg=15
## apse         0.015  0.023  0.025  0.057  0.054
```

```
## var_mutilde  0.006  0.013  0.014  0.040  0.041
## bias2        0.009  0.011  0.011  0.017  0.013
## var_y        0.000  0.000  0.000  0.000  0.000
```

- Let's visualize these results

```
plot(degrees, apse_vals[3,], xlab="Degree", ylab="", type='l',
      ylim=c(0, 0.026), col="firebrick", lwd=2 )
lines(degrees, apse_vals[2,], xlab="Degree", ylab="", col="steelblue", lwd=2 )
lines(degrees, apse_vals[1,], col="purple", lwd=2)
lines(degrees, apse_vals[4,], col="black", lwd=2)
```



* Bias-Variance Trade Off * This is the phenomenon whereby APSE is large for uncomplicated models due to increased bias, and APSE is large for overly complicated models due to increased variance. The optimal model is one which balances bias and variance

- **Recall** that we can also calculate the APSE using only the ~~training~~ test data $\mathcal{T}$. The function `apse_test` defined below does exactly this.
  - The difference between this function and `apse_all` is that `apse_all` performed the APSE calculation on the entire population $\mathcal{P}$

```
apse_test <- function(Ssamples, Tsamples, complexity, tau){
  ## average over the samples S
  ##
  N_S <- length(Ssamples)
  muhats <- lapply(Ssamples,
                   FUN=function(sample) getmuhat(sample, complexity)
  )
  ## get the average of these, mubar
```

```
  mubar <- getmubar(muhats)

  rowMeans(sapply(1:N_S,
                  FUN=function(j){
                    T_j <- Tsamples[[j]]
                    muhat <- muhats[[j]]
                    ## Take care of any NAs
                    T_j <- na.omit(T_j)
                    y <- T_j$y
                    x <- T_j$x
                    tau_x <- tau(x)
                    muhat_x <- muhat(x)
                    mubar_x <- mubar(x)

                    apse         <- (y - muhat_x)
                    bias2        <- (mubar_x - tau_x)
                    var_mutilde <-  (muhat_x - mubar_x)
                    var_y <- (y - tau_x)

                    ## Put them together and square them
                    squares <- rbind(apse, var_mutilde, bias2, var_y)^2
                    rowMeans(squares)
                  }
  ))
}
```

- Applying this function to our example (and increasing the degrees considered) yields the following results:

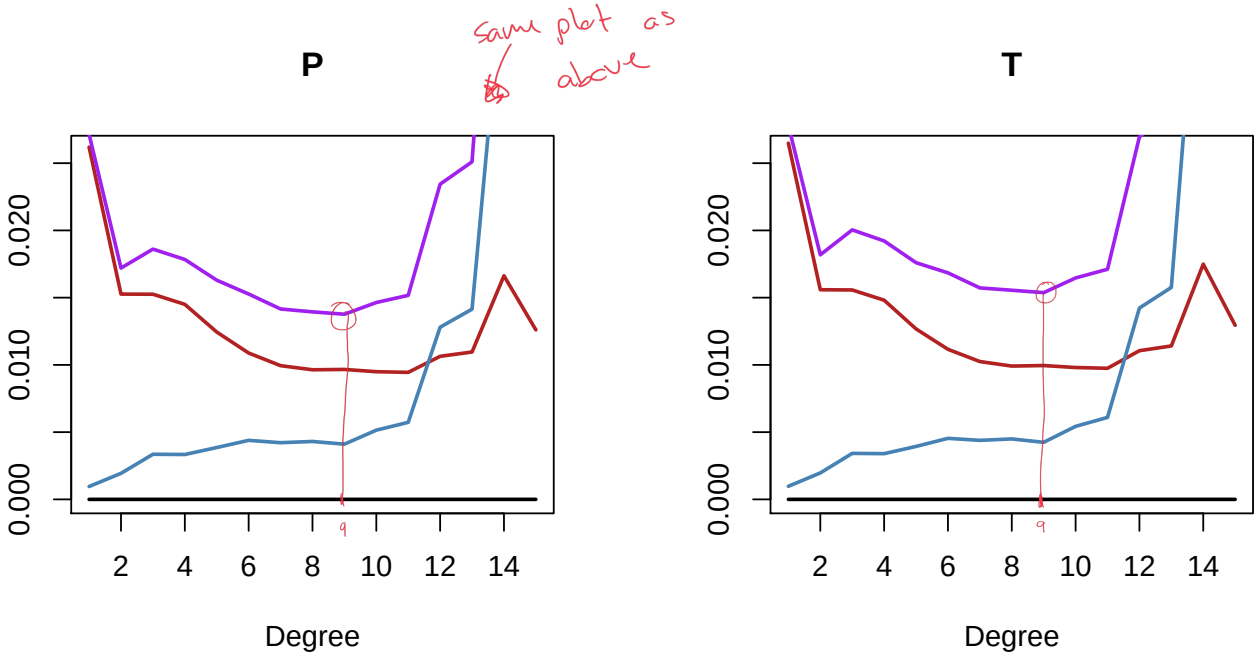
```
degrees <- 1:15
tau.annual = gettauFun(pop, xnam, ynam)

apse_vals_test <-  sapply(degrees, FUN = function(deg){
              apse_test(Ssam, Tsam, complexity = deg, tau = tau.annual)
    } )

colnames(apse_vals_test) =  paste("deg=", degrees, sep="")

round(apse_vals_test,3)
```

```
##             deg=1 deg=2 deg=3 deg=4 deg=5 deg=6 deg=7 deg=8 deg=9 deg=10
## apse        0.028 0.018 0.020 0.019 0.018 0.017 0.016 0.016 0.015  0.016
## var_mutilde 0.001 0.002 0.003 0.003 0.004 0.005 0.004 0.004 0.004  0.005
## bias2       0.026 0.016 0.016 0.015 0.013 0.011 0.010 0.010 0.010  0.010
## var_y       0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000  0.000
##             deg=11 deg=12 deg=13 deg=14 deg=15
## apse         0.017  0.027  0.029  0.066  0.063
## var_mutilde  0.006  0.014  0.016  0.045  0.047
## bias2        0.010  0.011  0.011  0.017  0.013
## var_y        0.000  0.000  0.000  0.000  0.000
```

- Note that the APSE calculated using only the test samples does not have the same additive property.

- Let's visualize these and the previous results together

```
par(mfrow=c(1,2))

plot( degrees, apse_vals[3,], xlab="Degree", ylab="", type='l',
      ylim=c(0, 0.026), col="firebrick", lwd=2 , main = "P")
lines(degrees, apse_vals[2,], xlab="Degree", ylab="", col="steelblue", lwd=2 )
lines(degrees, apse_vals[1,], col="purple", lwd=2)
lines(degrees, apse_vals[4,], col="black", lwd=2)

plot( degrees, apse_vals_test[3,], xlab="Degree", ylab="", type='l',
      ylim=c(0, 0.026), col="firebrick", lwd=2 , main = "T")
lines(degrees, apse_vals_test[2,], xlab="Degree", ylab="", col="steelblue", lwd=2 )
lines(degrees, apse_vals_test[1,], col="purple", lwd=2)
lines(degrees, apse_vals_test[4,], col="black", lwd=2)
```



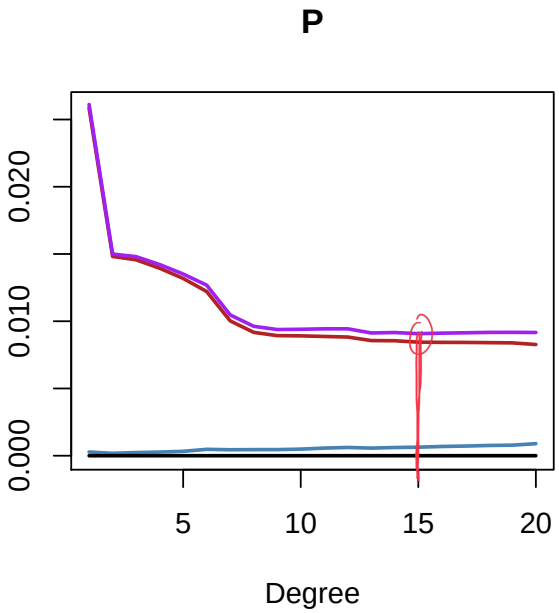
- Comparision of the APSE shows that both approaches identify the same degree (9) as being optimal.

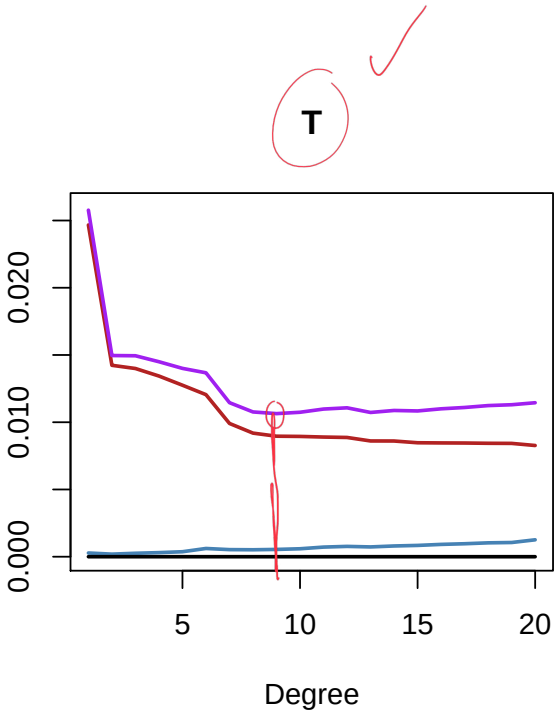
Sensitivity to n and N_S

- It is worth investigating whether varying N_S and n drastically changes our conclusions
  - Let's now try taking $N_S = 25$ samples of size $n = 100$ from the Global Temperature population.

```
N_S <- 25
xnam <- "YEAR"
ynam <- "ANNUAL"
pop  <- temperature
```

```r
n    <- 100


set.seed(341)  # for reproducibility
samples <- lapply(1:N_S, FUN= function(i){getSampleComp(pop, n)})
Ssam    <- lapply(samples, FUN= function(Si){getXYSample(xnam, ynam, Si, pop)})
Tsam    <- lapply(samples, FUN= function(Si){getXYSample(xnam, ynam, !Si, pop)})

degrees <- 1:20
tau.annual = gettauFun(pop, xnam, ynam)

apse_vals <-  sapply(degrees, FUN = function(deg){
              apse_all(Ssam, Tsam, complexity = deg, tau = tau.annual)
    } )

apse_vals_test <-  sapply(degrees, FUN = function(deg){
              apse_test(Ssam, Tsam, complexity = deg, tau = tau.annual)
    } )

colnames(apse_vals)      =  paste("deg=", degrees, sep="")
colnames(apse_vals_test) =  paste("deg=", degrees, sep="")

par(mfrow=c(1,2))
plot( degrees, apse_vals[3,], xlab="Degree", ylab="", type='l',
      ylim=c(0, 0.026), col="firebrick", lwd=2 , main = "P")
lines(degrees, apse_vals[2,], xlab="Degree", ylab="", col="steelblue", lwd=2 )
lines(degrees, apse_vals[1,], col="purple", lwd=2)
lines(degrees, apse_vals[4,], col="black", lwd=2)

plot( degrees, apse_vals_test[3,], xlab="Degree", ylab="", type='l',
      ylim=c(0, 0.026), col="firebrick", lwd=2 , main = "T")
lines(degrees, apse_vals_test[2,], xlab="Degree", ylab="", col="steelblue", lwd=2 )
lines(degrees, apse_vals_test[1,], col="purple", lwd=2)
lines(degrees, apse_vals_test[4,], col="black", lwd=2)
```

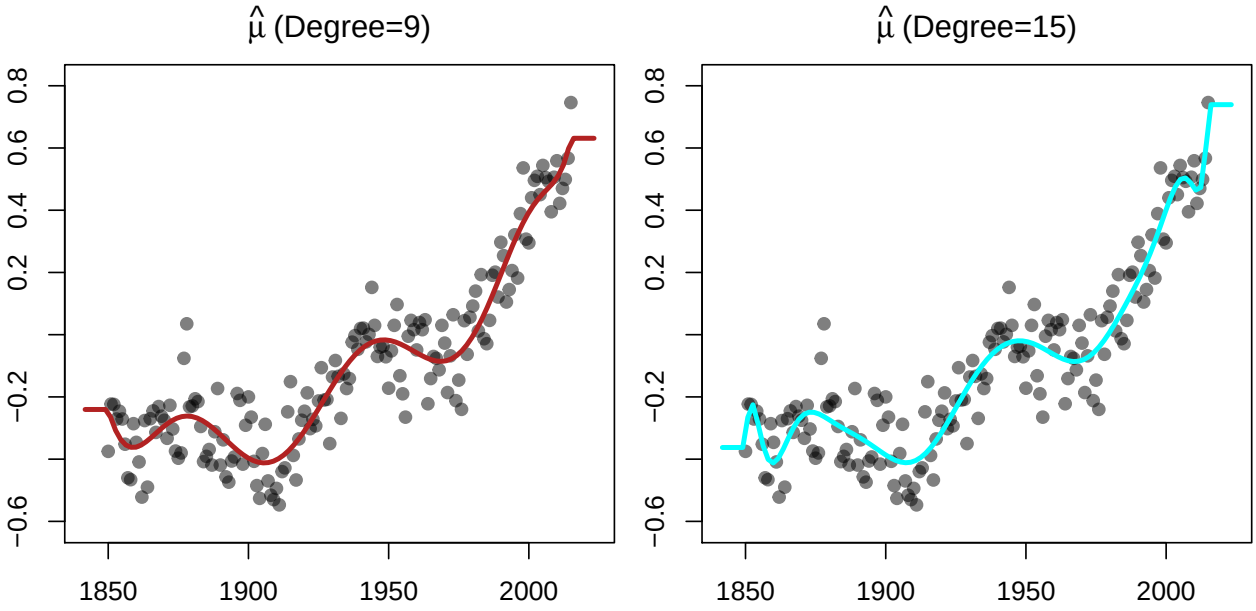




- Can you guess which colour is which by just looking at the trends?
  - black: $Ave_{\mathbf{x}}\left(Var[Y|\mathbf{x}]\right)$
  - blue: $Var[\widehat{\mu}]$
  - red: $Bias^2[\widehat{\mu}]$
  - purple: $APSE$

- The two APSE curves shown here identify different degrees as being optimal
  - Using the whole population (both the samples and test sets) we pick degree = 15
  - Using just the test sets we pick degree = 9

- Let's visualize both of these polynomials fit to all of the data.



- Do the results differ dramatically? No
- Which degree should you feel more confident choosing? 9, to protect against over fitting.

5.2.4 Example: Loblolly Pine Data

- Here we set $N_S = 25$ and $n = 40$.

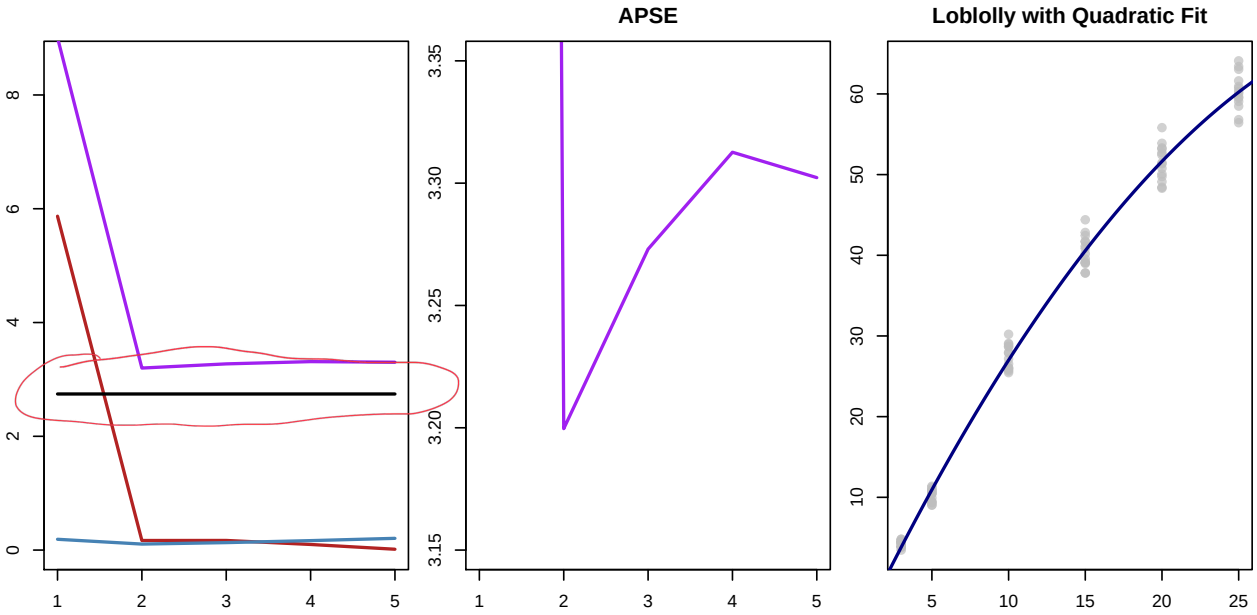
```
N_S <- 25
xnam <- "age"
ynam <- "height"
pop  <- Loblolly
n    <- 40

set.seed(341)  # for reproducibility
samples <- lapply(1:N_S, FUN= function(i){getSampleComp(pop, n)})
Ssam    <- lapply(samples, FUN= function(Si){getXYSample(xnam, ynam, Si, pop)})
Tsam    <- lapply(samples, FUN= function(Si){getXYSample(xnam, ynam, !Si, pop)})
```

- Then we fit polynomials of different degrees to each the $N_S = 25$ samples and calculate the APSE.

```
degrees <- 1:5
tau.age = gettauFun(pop, xnam, ynam)

set.seed(341)
apse_vals <-  sapply(degrees, FUN = function(deg){
              apse_test(Ssam, Tsam, complexity = deg, tau = tau.age)
    } )
```

```r
colnames(apse_vals) =  paste("deg=", degrees, sep="")
round(apse_vals,4)
```

```
##                deg=1  deg=2  deg=3  deg=4  deg=5
## apse         9.0085 3.1996 3.2730 3.3126 3.3023
## var_mutilde 0.1897 0.1055 0.1310 0.1662 0.2065
## bias2        5.8694 0.1684 0.1692 0.0986 0.0150
## var_y        2.7440 2.7440 2.7440 2.7440 2.7440
```

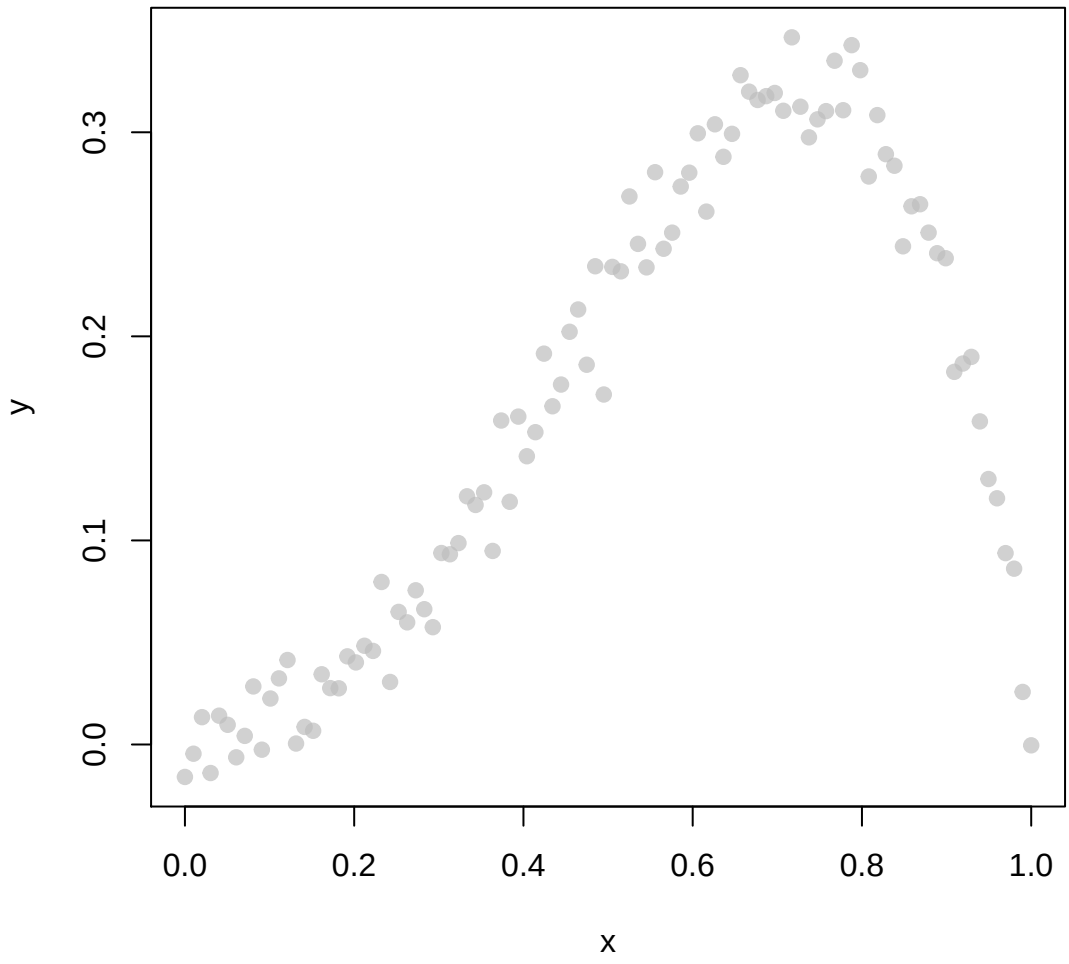


5.2.5 Example: Fake Data

- Here we simulate a fake population (of size $N = 100$) according to

$$y_i = x_i^2 - x_i^5 + r_i$$

```r
set.seed(341)
x = seq(0,1, length.out=100)
y = x^2 - x^5 + rnorm(100, sd=.015)
plot(x, y, col=adjustcolor("grey", 0.7), pch=19)
```



$N_S = 25$ **and** $n = 25$

- We will start with $N_S = 25$ samples, each of size $n = 25$.

```
datax = data.frame(x=x,y=y)
N_S  <- 25
xnam <- "x"
ynam <- "y"
pop  <- datax
n    <- 25

set.seed(341)
samples <- lapply(1:N_S, FUN= function(i){getSampleComp(pop, n)})
Ssam    <- lapply(samples, FUN= function(Si){getXYSample(xnam, ynam, Si, pop)})
Tsam    <- lapply(samples, FUN= function(Si){getXYSample(xnam, ynam, !Si, pop)})
```

- Calculate the APSE

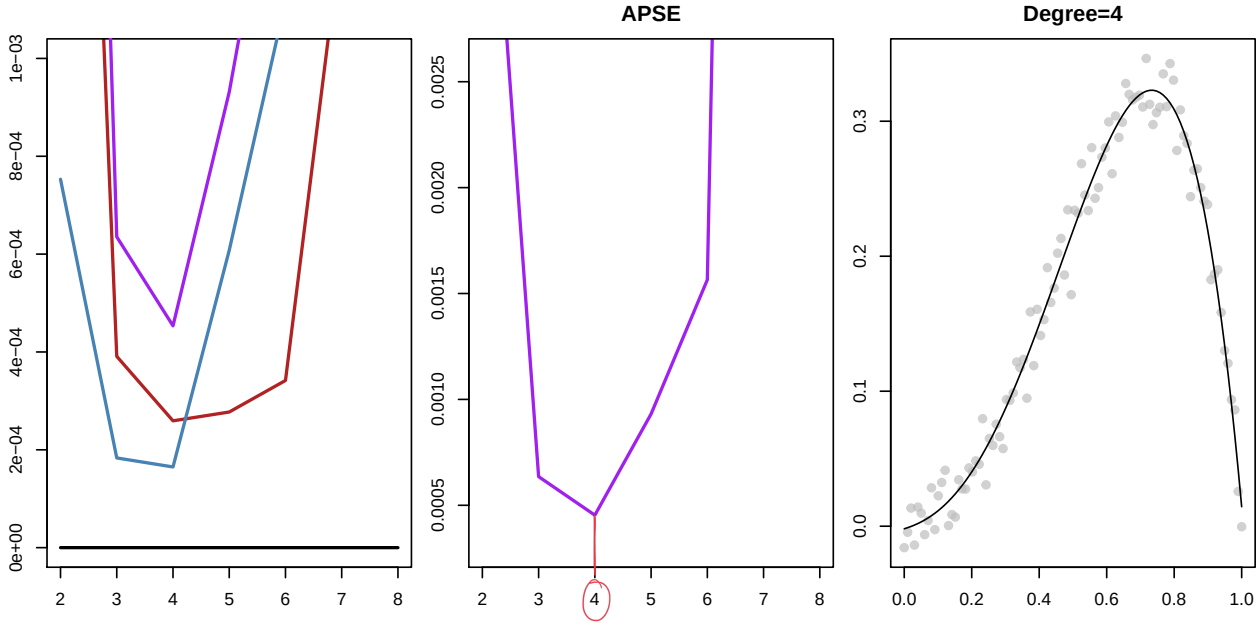
```
degrees <- 2:8
tau.x = gettauFun(pop, xnam, ynam)

apse_vals <-  sapply(degrees, FUN = function(deg){
```

```
            apse_test(Ssam, Tsam, complexity = deg, tau = tau.x)
    } )

colnames(apse_vals) =  paste("deg=", degrees, sep="")
round(apse_vals,5)
```

```
##                deg=2   deg=3   deg=4   deg=5   deg=6   deg=7   deg=8
## apse         0.00429 0.00064 0.00045 0.00093 0.00156 0.01459 0.07619
## var_mutilde 0.00075 0.00018 0.00016 0.00061 0.00112 0.01255 0.07149
## bias2        0.00318 0.00039 0.00026 0.00028 0.00034 0.00126 0.00275
## var_y        0.00000 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000
```

- Plot the results

```
par(mfrow=c(1,3), mar=2.5*c(1,1,1,0.1))

plot( degrees, apse_vals[3,], xlab="Degree", ylab="", type='l',
      ylim=c(0, 0.001),
      col="firebrick", lwd=2 )
lines(degrees, apse_vals[2,], xlab="Degree", ylab="", col="steelblue", lwd=2 )
lines(degrees, apse_vals[1,], col="purple", lwd=2)
lines(degrees, apse_vals[4,], col="black", lwd=2)

plot( degrees, apse_vals[1,], xlab="Degree", ylab="", type='l',
      col="purple", lwd=2, ylim=c(0.00030, 0.00261), main="APSE" )

plot(datax$x, datax$y, xlab="x", ylab="y", col=adjustcolor("grey", 0.7), pch=19, main="Degree=4")

samL = rep(TRUE, nrow(datax))
sample.Data <- getXYSample(xnam, ynam, samL, datax)
mu.age = getmuhat(sample.Data, complexity = 4)
curve(mu.age, 0, 1, add=TRUE, n=100)
```



$N_S = 25$ **and** $n = 75$

- Let us keep $N_S = 25$ but increase the sample size from $n = 25$ to $n = 75$.

```
datax = data.frame(x=x,y=y)
N_S  <- 25
xnam <- "x"
ynam <- "y"
pop  <- datax
n    <- 75

set.seed(341)
samples <- lapply(1:N_S, FUN= function(i){getSampleComp(pop, n)})
Ssam    <- lapply(samples, FUN= function(Si){getXYSample(xnam, ynam, Si, pop)})
Tsam    <- lapply(samples, FUN= function(Si){getXYSample(xnam, ynam, !Si, pop)})
```

- Calculate the APSE

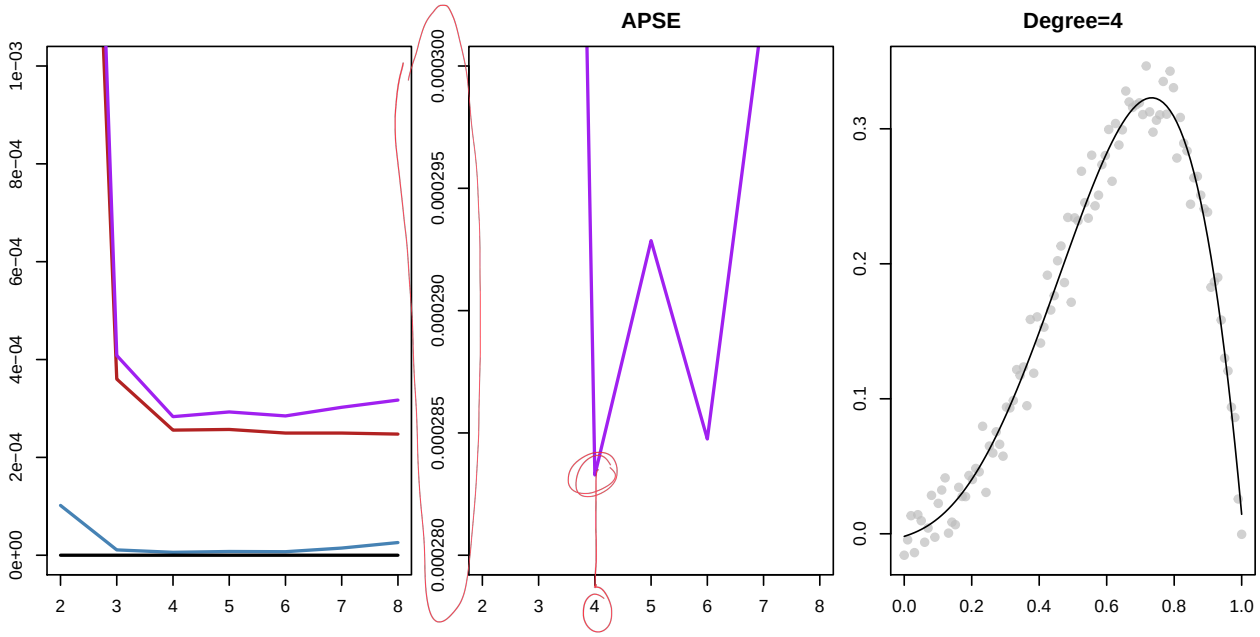
```
degrees <- 2:8
tau.x = gettauFun(pop, xnam, ynam)

apse_vals <-  sapply(degrees, FUN = function(deg){
              apse_test(Ssam, Tsam, complexity = deg, tau = tau.x)
    } )

colnames(apse_vals) =  paste("deg=", degrees, sep="")
round(apse_vals,6)
```

```
##                deg=2    deg=3    deg=4    deg=5    deg=6    deg=7    deg=8
## apse        0.003687 0.000408 0.000283 0.000293 0.000285 0.000302 0.000317
## var_mutilde 0.000102 0.000011 0.000006 0.000008 0.000007 0.000015 0.000026
## bias2       0.003219 0.000360 0.000256 0.000257 0.000250 0.000250 0.000248
## var_y       0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
```

- Plot the results

```
par(mfrow=c(1,3), mar=2.5*c(1,1,1,0.1))

plot( degrees, apse_vals[3,], xlab="Degree", ylab="", type='l',
      ylim=c(0, 0.001), col="firebrick", lwd=2 )
lines(degrees, apse_vals[2,], xlab="Degree", ylab="", col="steelblue", lwd=2 )
lines(degrees, apse_vals[1,], col="purple", lwd=2)
lines(degrees, apse_vals[4,], col="black", lwd=2)


plot( degrees, apse_vals[1,], xlab="Degree", ylab="", type='l',
      col="purple", lwd=2, ylim=c(0.00028, 0.00030), main="APSE" )


plot(datax$x, datax$y, xlab="x", ylab="y",
```

```
     col=adjustcolor("grey", 0.7), pch=19, main="Degree=4")


samL = rep(TRUE, nrow(datax))
sample.Data <- getXYSample(xnam, ynam, samL, datax)
mu.age = getmuhat(sample.Data, complexity = 4)
curve(mu.age, 0, 1, add=TRUE, n=100)
```



$N_S = 25$ **and** $n = 95$

- Again, we'll keep $N_S = 25$ but increase the sample size from $n = 75$ to $n = 95$.

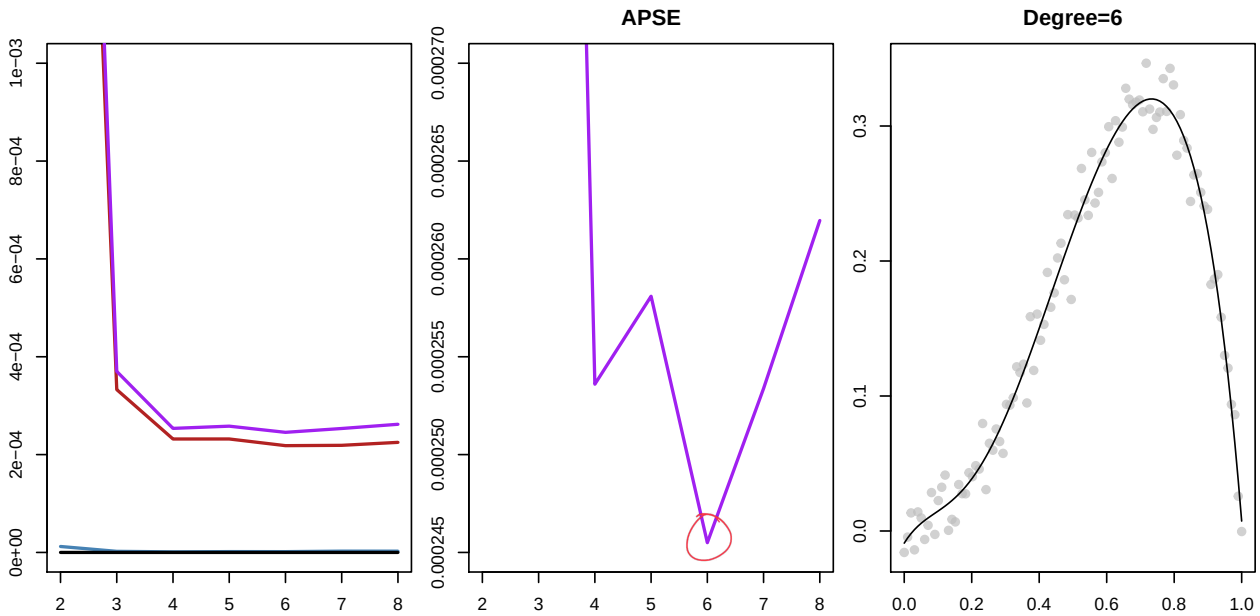
```
datax = data.frame(x=x,y=y)
N_S  <- 25
xnam <- "x"
ynam <- "y"
pop  <- datax
n    <- 95

set.seed(341)
samples <- lapply(1:N_S, FUN= function(i){getSampleComp(pop, n)})
Ssam    <- lapply(samples, FUN= function(Si){getXYSample(xnam, ynam, Si, pop)})
Tsam    <- lapply(samples, FUN= function(Si){getXYSample(xnam, ynam, !Si, pop)})
```

```
degrees <- 2:8
tau.x = gettauFun(pop, xnam, ynam)

apse_vals <-  sapply(degrees, FUN = function(deg){
              apse_test(Ssam, Tsam, complexity = deg, tau = tau.x)
    } )
```

```
colnames(apse_vals) =  paste("deg=", degrees, sep="")
round(apse_vals,5)
```

```
##                deg=2   deg=3   deg=4   deg=5   deg=6   deg=7   deg=8
## apse         0.00340 0.00037 0.00025 0.00026 0.00025 0.00025 0.00026
## var_mutilde 0.00001 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000
## bias2        0.00314 0.00033 0.00023 0.00023 0.00022 0.00022 0.00023
## var_y        0.00000 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000
```

```
par(mfrow=c(1,3), mar=2.5*c(1,1,1,0.1))

plot( degrees, apse_vals[3,], xlab="Degree", ylab="", type='l',
      ylim=c(0, 0.001), col="firebrick", lwd=2 )
lines(degrees, apse_vals[2,], xlab="Degree", ylab="", col="steelblue", lwd=2 )
lines(degrees, apse_vals[1,], col="purple", lwd=2)
lines(degrees, apse_vals[4,], col="black", lwd=2)


plot( degrees, apse_vals[1,], xlab="Degree", ylab="", type='l',
      col="purple", lwd=2, ylim=c(0.000245, 0.00027), main="APSE" )


plot(datax$x, datax$y, xlab="x", ylab="y",
     col=adjustcolor("grey", 0.7), pch=19, main="Degree=6")


samL = rep(TRUE, nrow(datax))
sample.Data <- getXYSample(xnam, ynam, samL, datax)
mu.age = getmuhat(sample.Data, complexity = 6)
curve(mu.age, 0, 1, add=TRUE, n=100)
```



* Not much difference between degree 4 and 6, so let's choose 4.