Nonparametric Models

Fitting Models to Data, Not Data to Models

Model Fitting Series - With Applications in R

Jesse Ghashti

Source code by Jesse Ghashti

December 2, 2025

Centre for Scholarly Communication
The University of British Columbia | Okanagan Campus | Syilx Okanagan Nation Territory

Workshop Series Overview

Session	Topic	Date/Time
1	Simple Linear Regression	Oct 7, 9:00 AM
2	Fitting Linear Models in R	Oct 8, 10:30 AM
3	Multiple Linear Regression in R	Oct 16, 4:00 PM
4	Interaction Terms & Hierarchical Linear Models	Oct 21, 11:00 AM
5	Generalized Linear Models	Oct 23, 4:00 PM
6	Generalized Additive Models (GAMs)	Oct 28, 11:00 AM
7	Interpreting & Predicting from GAMs	Oct 29, 10:30 AM
8	Hierarchical GAMs	Nov 4, 12:00 PM
9	Penalized Models	Nov 18, 11:00 AM
10	Survival Models	Nov 25, 11:00 AM
11	Nonparametric Models	Dec 2, 11:00 AM

Today: Nonparametric Regression and Prediction

Today we will...

- Explore the `airquality` dataset for ground-level ozone prediction
- Implement k-Nearest Neighbours (kNN) regression with cross-validation
- Fit kernel regression using local polynomial methods
- Visualize sliced effects to understand model behaviour

We will use these packages...

```
# install.packages(c("ggplot2", "dplyr", "FNN", "np", "patchwork"))
library(ggplot2)
library(dplyr)
library(FNN)        # kNN regression
library(np)         # kernel regression
library(tidyr)
library(mgcv)
```

New York Air Quality (1973)

Prediction Goal

We analyze the `airquality` dataset to predict ground-level ozone from meteorological variables:

- Response: Ozone concentration (ppb)
- Predictors:
  - Solar.R: Solar radiation (Langleys)
  - Wind: Wind speed (mph)
  - Temp: Temperature (°F)
  - Month: Month of observation
  - Day: Day of month
- Challenge: Nonlinear relationships between predictors and ozone
- Goal: Compare parametric and nonparametric approaches

Loading and Cleaning the Data

Data preparation

```r
## 1. load+clean data
data("airquality")
aq <- airquality %>%
  select(Ozone, Solar.R, Wind, Temp, Month, Day) %>%
  na.omit()
str(aq)
summary(aq)
```

Ozone	Solar.R	Wind	Temp	Month	Day
Min. : 1.0	Min. : 7.0	Min. : 2.30	Min. :57.00	Min. :5.000	Min. : 1.00
1st Qu.: 18.0	1st Qu.:113.5	1st Qu.: 7.40	1st Qu.:71.00	1st Qu.:6.000	1st Qu.: 9.00
Median : 31.0	Median :207.0	Median : 9.70	Median :79.00	Median :7.000	Median :16.00
Mean : 42.1	Mean :184.8	Mean : 9.94	Mean :77.79	Mean :7.216	Mean :15.95
3rd Qu.: 62.0	3rd Qu.:255.5	3rd Qu.:11.50	3rd Qu.:84.50	3rd Qu.:9.000	3rd Qu.:22.50
Max. :168.0	Max. :334.0	Max. :20.70	Max. :97.00	Max. :9.000	Max. :31.00

Exploratory Data Analysis: Correlations

Correlation analysis

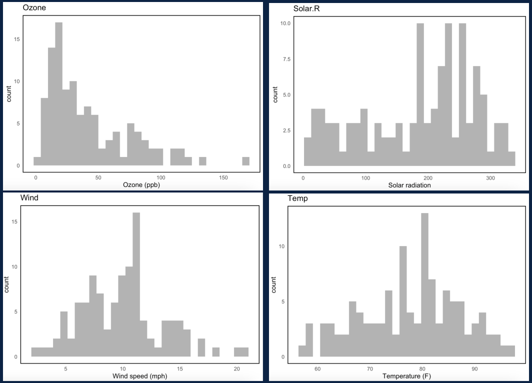
```r
## 2. exploring data
## correlation matrix for continuous vars
cor(aq[, c("Ozone", "Solar.R",
           "Wind", "Temp")])
```

	Ozone	Solar.R	Wind	Temp
Ozone	1.0000000	0.3483417	-0.6124966	0.6985414
Solar.R	0.3483417	1.0000000	-0.1271835	0.2940876
Wind	-0.6124966	-0.1271835	1.0000000	-0.4971897
Temp	0.6985414	0.2940876	-0.4971897	1.0000000

Univariate Distributions

Creating histograms

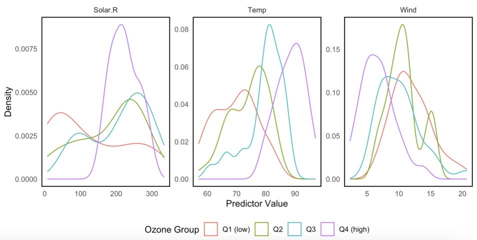
```r
## histograms for each variable
pOZ   <- ggplot(aq, aes(x = Ozone))+
  geom_histogram(bins = 30, fill = "grey70")+
  theme_minimal() +
  labs(title = "Ozone", x = "Ozone (ppb)")+
  theme(panel.border = element_rect(NA, "black", 1),
        panel.grid = element_blank())
pSOL  <- ggplot(aq, aes(x = Solar.R))+
  geom_histogram(bins = 30, fill = "grey70")+
  theme_minimal() +
  labs(title = "Solar.R", x = "Solar radiation")+
  theme(panel.border = element_rect(NA, "black", 1),
        panel.grid = element_blank())
pWIND <- ggplot(aq, aes(x = Wind))+
  geom_histogram(bins = 30, fill = "grey70")+
  theme_minimal() +
  labs(title = "Wind", x = "Wind speed (mph)")+
  theme(panel.border = element_rect(NA, "black", 1),
        panel.grid = element_blank())
pTEMP <- ggplot(aq, aes(x = Temp))+
  geom_histogram(bins = 30, fill = "grey70")+
  theme_minimal() +
  labs(title = "Temp", x = "Temperature (F)")+
  theme(panel.border = element_rect(NA, "black", 1),
        panel.grid = element_blank())
```



Predictor Distributions by Ozone Level

Quartile analysis

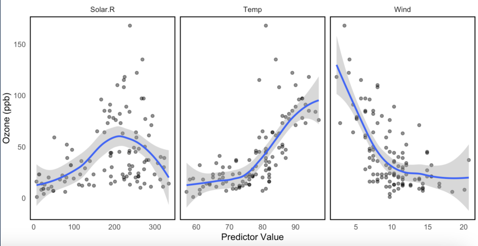
```r
# ozone quartile groups
aqDENSITY <- aq %>%
  mutate(Ozone_group = cut(Ozone,
        breaks = quantile(Ozone,
        probs = seq(0, 1, by = 0.25),
        na.rm = TRUE),
      include.lowest = TRUE,
      labels = c("Q1 (low)", "Q2", "Q3", "Q4 (high)")))
predVARS <- c("Solar.R", "Wind", "Temp")
aqLONG <- aqDENSITY %>%
  pivot_longer(
    cols = all_of(predVARS),
    names_to = "variable",
    values_to = "value")
# faceted density plots of predictors
ggplot(aqLONG, aes(x = value, colour = Ozone_group)) +
  geom_density() +
  facet_wrap(~ variable, scales = "free") +
  theme_minimal() +
  labs(title = "Predictor distributions by Ozone level",
    x = "Predictor Value",
    y = "Density",
    colour = "Ozone Group"
  )+
  theme(panel.border = element_rect(NA, "black", 1),
        panel.grid = element_blank())
```



Ozone vs Predictors: Nonlinear Relationships

Loess smoothers

```
aqLONGscatter <- aq %>%
  pivot_longer(
    cols = all_of(predVARS),
    names_to = "variable",
    values_to = "value"
  )
ggplot(aqLONGscatter, aes(x = value, y = Ozone)) +
  geom_point(alpha = 0.5) +
  geom_smooth(method = "loess", se = TRUE) +
  facet_wrap(~ variable, scales = "free_x") +
  theme_minimal() +
  labs(
    title = "Ozone vs Predictors w/ Smoothers",
    x = "Predictor Value",
    y = "Ozone (ppb)"
  )+
  theme(panel.border = element_rect(NA, "black", 1),
        panel.grid = element_blank())
```



Creating Training and Test Sets

70-30 split

```
## 3. Train/test split
set.seed(112025)
n       <- nrow(aq)
trainIDX <- sample(seq_len(n), size = round(0.7*n))
train <- aq[trainIDX, ]
test  <- aq[-trainIDX, ]
# nrow(train); nrow(test)
## 4. utilities
rmse <- function(y, yhat) sqrt(mean((y-yhat)^2))
r2 <- function(y, yhat) 1-sum((y-yhat)^2)/sum((y-mean(y))^2)
```

Recall some model evaluation metrics

- RMSE: Root Mean Squared Error - lower is better
- R^2: Proportion of variance explained - higher is better
- MAE: Mean Absolute Error - robust to outliers

Baseline: Linear Regression

Parametric baseline

$$\text{Ozone}_i = \beta_0 + \beta_1 \text{Wind}_i + \beta_2 \text{Temp}_i + \beta_3 \text{Solar.R}_i + \beta_4 \text{Month}_i + \epsilon_i$$

Assumptions:

- Linear relationships between predictors and response
- Constant variance (homoscedasticity)
- Independent errors
- Normal residuals

Fitting Linear Regression

Linear model

```r
## 5. baseline: Linear regression
## Ozone ~ Wind+Temp+Solar.R+Month
lmMod <- lm(Ozone ~ Wind+Temp+Solar.R+factor(Month),
            data = train)
summary(lmMod)
## month is not statistically significant
## so remove it moving forward

lmTrainPred <- predict(lmMod, newdata = train)
lmTestPred  <- predict(lmMod, newdata = test)

data.frame(
  dataset = c("train", "test"),
  RMSE    = c(rmse(train$Ozone, lmTrainPred),
              rmse(test$Ozone,  lmTestPred)),
  R2      = c(r2(train$Ozone, lmTrainPred),
              r2(test$Ozone,  lmTestPred))
)
```

	dataset	RMSE	R2
1	train	17.06931	0.6695807
2	test	26.34482	0.5497042

```
Call:
lm(formula = Ozone ~ Wind + Temp + Solar.R + factor(Month), data

Residuals:
    Min      1Q  Median      3Q     Max
-34.379 -11.902  -2.241   9.866  49.954

Coefficients:
                Estimate Std. Error t value Pr(>|t|)
(Intercept)    -96.37417   27.78875  -3.468 0.000901 ***
Wind            -2.26732    0.71550  -3.169 0.002269 **
Temp             2.05649    0.36050   5.705 2.58e-07 ***
Solar.R          0.05328    0.02512   2.121 0.037430 *
factor(Month)6 -19.60412    9.20737  -2.129 0.036759 *
factor(Month)7 -11.98332    7.81321  -1.534 0.129605
factor(Month)8  -9.20317    8.95971  -1.027 0.307875
factor(Month)9 -15.68992    6.62717  -2.368 0.020676 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 18.02 on 70 degrees of freedom
Multiple R-squared:  0.6696,	Adjusted R-squared:  0.6365
F-statistic: 20.26 on 7 and 70 DF,  p-value: 1.374e-14
```

k-Nearest Neighbours Regression

Model + Prediction

Prediction rule:

$$\hat{y}_{\text{new}} = \frac{1}{k} \sum_{i \in N_k(\mathbf{x}_{\text{new}})} y_i, \qquad N_k(\mathbf{x}_{\text{new}}) = \{i : ||\mathbf{x}_{\text{new}} - \mathbf{x}_i|| \leq d_k(\mathbf{x}_{\text{new}})\}$$

Where $N_k(\mathbf{x}_{\text{new}})$ are the k nearest neighbours to $\mathbf{x}_{\text{new}}$ in feature space.

- No explicit model - purely data-driven
- Local averaging in feature space
- Requires scaling of predictors
- Choice of k controls bias-variance tradeoff:
  - Small k: flexible, low bias, high variance
  - Large k: smooth, high bias, low variance

kNN: Feature Scaling

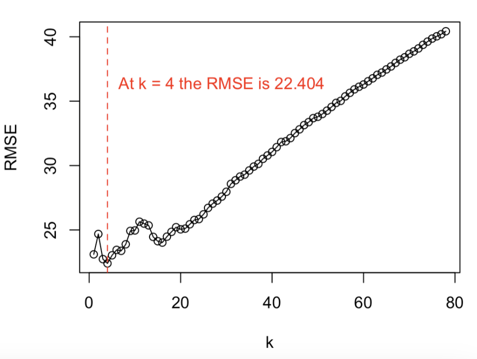
Standardizing predictors

```r
## 6. k-Nearest Neighbour (kNN) regression
## kNN needs numeric matrix; scale predictors
predVARS <- c("Wind", "Temp", "Solar.R")

scalingFun <- function(df, cols) {
  mu <- sapply(df[, cols, drop = FALSE], mean)
  sd <- sapply(df[, cols, drop = FALSE], sd)
  x  <- scale(df[, cols, drop = FALSE],
              center = mu, scale = sd)
  list(x = x, mu = mu, sd = sd)
}

scaleTrain <- scalingFun(train, predVARS)
xTrain  <- scaleTrain$x
xTest <- scale(test[, predVARS, drop = FALSE],
               center = scaleTrain$mu,
               scale  = scaleTrain$sd)
yTrain <- train$Ozone
yTest  <- test$Ozone
```

Finding Best k with Cross-Validation

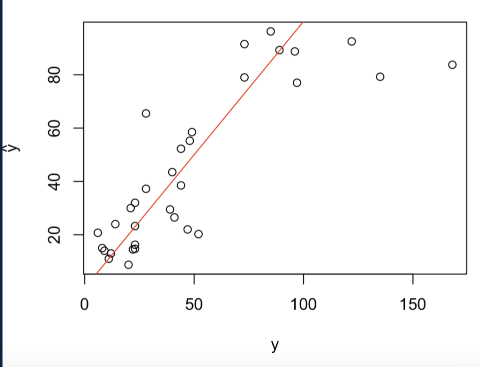
Grid search for k

```r
## line search k-values to find the best one
## to minimize RMSE
kGrid <- c(1:nrow(xTrain))
knnGridRes <- lapply(kGrid, function(k) {
  out <- knn.reg(train = xTrain,
                 test = xTest,
                 y = yTrain,
                 k = k)
  pred <- out$pred
  data.frame(
    k    = k,
    RMSE = rmse(yTest, pred),
    R2   = r2(yTest, pred)
  )
})
knnGridSum <- do.call(rbind, knnGridRes)

plot(knnGridSum$k, knnGridSum$RMSE,
     type = "l", xlab = "k", ylab = "RMSE")
points(knnGridSum$k, knnGridSum$RMSE)
abline(v = knnGridSum$k[which.min(knnGridSum$RMSE)],
       lty = "dashed", col = "red")
```



kNN Predictions

```r
plot(knnGridSum$k, knnGridSum$R2,
     type = "l", xlab = "k", ylab = "R2")
points(knnGridSum$k, knnGridSum$R2)
abline(v = knnGridSum$k[which.max(knnGridSum$R2)],
       lty = "dashed", col = "red")
text(x = knnGridSum$k[which.max(knnGridSum$R2)]+8,
     y = mean(c(max(knnGridSum$R2),
              min(knnGridSum$R2))),
     labels = paste0("At k = ",
                     knnGridSum$k[which.max(knnGridSum$R2)],
                     " the R2 is ",
                     round(max(knnGridSum$R2),3)),
     col = "red")

## choose best k by RMSE
bestkIDX <- which.min(knnGridSum$RMSE)
bestK    <- knnGridSum$k[bestkIDX]
bestK

## refit best k to get predictions
knnMod <- knn.reg(train = xTrain, test = xTest,
                  y = yTrain, k = bestK)
knnTestPred <- knnMod$pred
plot(yTest, knnTestPred, xlab="y",
        ylab=expression(hat(y)))
abline(a = 0, b = 1, col = "red")
```



Kernel Regression

Local polynomial regression

Estimator

$$\hat{y}(\mathbf{x}) = \frac{\sum_{i=1}^{n} K_h(\mathbf{x} - \mathbf{x}_i) y_i}{\sum_{i=1}^{n} K_h(\mathbf{x} - \mathbf{x}_i)}$$

Where K_h is a kernel function with bandwidth h.

- Weighted local averaging (as opposed to uniform weighting in kNN)
- Bandwidth selection important (like k in kNN)
- Can use different kernels: Gaussian, Epanechnikov, etc.
- Automatic bandwidth selection via cross-validation

Smooth predictions, handles mixed continuous/categorical data, provides gradients

What are Kernels?

Kernel functions

A kernel is a weighting function that assigns importance to observations based on their distance from a target point.

- Give more weight to nearby observations
- Less weight to distant observations
- Create smooth local averages

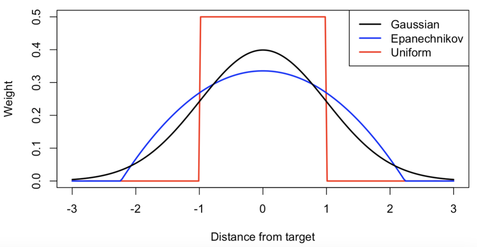
Common kernel functions:

- **Gaussian:** $K(u) = \frac{1}{\sqrt{2\pi}} e^{-\frac{u^2}{2}}$ (smooth, infinite support)
- **Epanechnikov:** $K(u) = \frac{3}{4}(1 - u^2) \cdot \mathbb{I}(|u| \leq 1)$ (optimal in theory)
- **Uniform:** $K(u) = \frac{1}{2} \cdot \mathbb{I}(|u| \leq 1)$ (simple box)

Visualizing Different Kernels

Kernel shapes

```r
# Comparing kernel functions
x <- seq(-3, 3, length.out = 200)
# Gaussian kernel
gaussian <- dnorm(x)
# Epanechnikov kernel
epanech <- ifelse(abs(x) <= sqrt(5),
                  3/(4*sqrt(5)) * (1 - x^2/5),
                  0)
# Uniform kernel
uniform <- ifelse(abs(x) <= 1, 0.5, 0)
plot(x, uniform, type = "l", lwd = 2,
     xlab = "Distance from target",
     ylab = "Weight",
     main = "Kernel Functions",
     col = "red") +
lines(x, epanech, col = "blue", lwd = 2) +
lines(x, gaussian, col = "black", lwd = 2) +
legend("topright",
       c("Gaussian", "Epanechnikov", "Uniform"),
       col = c("black", "blue", "red"),
       lwd = 2)
```



Understanding Bandwidth

The bandwidth parameter

Bandwidth (h) controls the width of the kernel window:

$$K_h(u) = \frac{1}{h} K\left(\frac{u}{h}\right)$$

Small bandwidth $h \to 0$

- Uses fewer neighbours (more local)
- Low bias but high variance
- Wiggly, potentially overfit curve

Large bandwidth $h \to \infty$

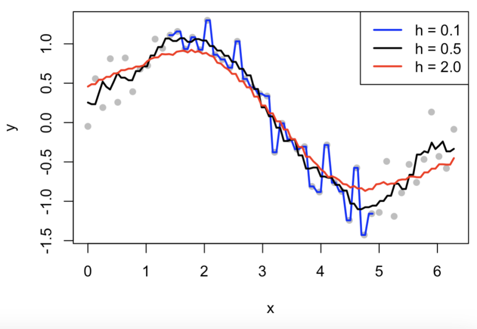
- Uses many neighbours (more global)
- High bias but low variance
- Smooth, potentially underfit curve

Cross-validation typically used to find optimal h

Bandwidth Effects on Smoothing

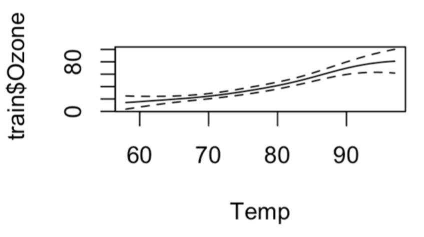
Three bandwidth choices

```
set.seed(11262025)
n <- 50
x <- seq(0, 2*pi, length.out = n)
y <- sin(x) + rnorm(n, 0, 0.3)
bwVALS <- c(0.1, 0.5, 2)
xGRID <- seq(0, 2*pi, length.out = 100)
plot(x, y, pch = 16, col = "gray",
     main = "Effect of Bandwidth",
     xlab = "x", ylab = "y")
colors <- c("blue", "black", "red")
for(i in 1:3) {
  h <- bwVALS[i]
  ySMOOTH <- ksmooth(x, y,
                     bandwidth = h,
                     x.points = xGRID)
  lines(ySMOOTH, col = colors[i], lwd = 2)
}
legend("topright",
       c("h = 0.1", "h = 0.5", "h = 2.0"),
       col = colors, lwd = 2)
```



Fitting Kernel Regression

```r
set.seed(112025)
npMod <- npreg(tydat  = train$Ozone, txdat  = train[, predVARS, drop = FALSE], regtype  = "ll",
               ckertype = "gaussian", newdata = xTest, y.eval = yTest, gradients = TRUE)
plot(npMod$bws, plot.errors.method = "bootstrap")
summary(npMod)
```




```
Regression Data: 78 training points, in 3 variable(s)
                  Wind     Temp  Solar.R
Bandwidth(s): 1.316773 3.415677 49870349

Kernel Regression Estimator: Local-Linear
Bandwidth Type: Fixed
Residual standard error: 10.86279
R-squared: 0.8673915

Continuous Kernel Type: Second-Order Gaussian
No. Continuous Explanatory Vars.: 3
```

Fitting Kernel Regression

```r
npTrainPred <- predict(npMod, exdat = train[, predVARS, drop = FALSE])
npTestPred  <- predict(npMod, exdat = test[,  predVARS, drop = FALSE])
data.frame(dataset = c("train", "test"),
RMSE = c(rmse(train$Ozone, npTrainPred), rmse(test$Ozone,  npTestPred)),
R2   = c(r2(train$Ozone, npTrainPred), r2(test$Ozone,  npTestPred)))
```

	dataset	RMSE	R2
1	train	10.86279	0.8661815
2	test	21.91970	0.6882714

Sliced Effects: Understanding Model Behaviour

Partial dependence

To understand how different models capture relationships:

- Fix other predictors at typical values, such as the median
- Vary one predictor across its range
- Compare predicted responses across models

Example: Ozone vs Temperature

- Hold Wind and Solar.R at median values
- Vary Temperature from min to max
- Plot predictions from LM, kNN, and kernel regression

This shows us

- Linear model: constant slope
- kNN: step-like function
- Kernel: smooth nonlinear curve

Computing Sliced Effects

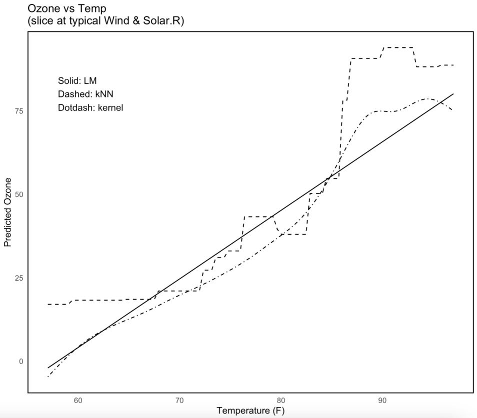
Temperature slice

```r
## 8. Sliced Effects
## Ozone vs Temp? Construct a slice
## at typical Wind and Solar.R and compare
medianWind <- median(aq$Wind)
medianSolar <- median(aq$Solar.R)
tempGrid <- seq(min(aq$Temp), max(aq$Temp),
                length.out = 100)
slideDF <- data.frame(
  Temp    = tempGrid,
  Wind    = medianWind,
  Solar.R = medianSolar,
  Month   = factor(7) # arbitrary factor for month
)
## lm predictions
slideDF$lmPreds <- predict(lmMod, newdata = slideDF)
## knn predictions
sliceX <- scale(slideDF[, predVARS, drop = FALSE],
                center = scaleTrain$mu,
                scale  = scaleTrain$sd)
sliceKNN <- knn.reg(
  train = xTrain,
  test  = sliceX,
  y     = yTrain,
  k     = bestK)
slideDF$knnPreds <- sliceKNN$pred
## kernel regression predictions
slideDF$npPreds <- predict(npMod,
        exdat = slideDF[, predVARS, drop = FALSE])
```

Visualizing Model Differences

Comparison plot

```r
ggplot(slideDF, aes(x = Temp)) +
  geom_line(aes(y = lmPreds),  linetype = "solid") +
  geom_line(aes(y = knnPreds), linetype = "dashed") +
  geom_line(aes(y = npPreds),  linetype = "dotdash") +
  theme_minimal() +
  labs(title = "Ozone vs Temp\n(slice at typical Wind & Solar.R)",
       x = "Temperature (F)",
       y = "Predicted Ozone") +
  annotate("text", x = min(tempGrid)+1,
          y = max(slideDF$lmPreds, na.rm=TRUE),
          label = "Solid: LM\nDashed: kNN\nDotdash: kernel",
          hjust = 0)+
  theme(panel.border = element_rect(NA, "black", 1),
        panel.grid = element_blank())
```



Bootstrap Confidence Bands for Effects

Uncertainty quantification

Why bootstrap for kNN?

- No analytical standard errors available
- Captures model uncertainty through resampling
- Provides confidence bands for predictions

Bootstrap procedure:

1. Sample with replacement from training data (B times)
2. Fit kNN on each bootstrap sample
3. Predict on evaluation grid
4. Calculate percentiles for confidence bands

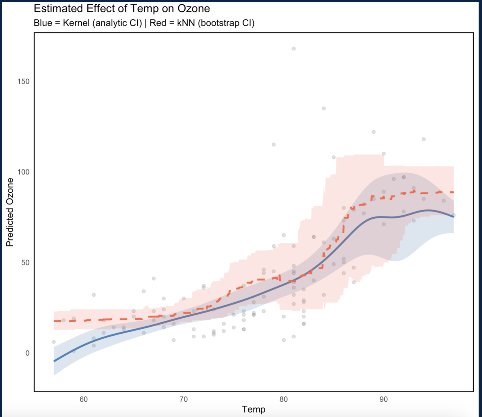
Kernel regression:

- Provides analytical standard errors
- Faster than bootstrap but may be less robust

Effect Curves with Confidence Bands

Bootstrap for kNN

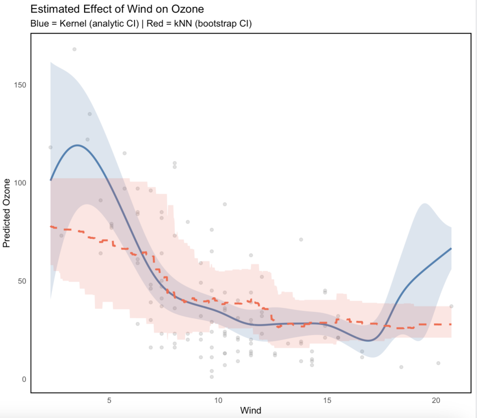
```
## 9. knn+kernel effect curves
vars <- c("Temp","Wind","Solar.R")
B   <- 1000 # number of bootstrap samples
for (v in vars) {
  grid <- seq(min(aq[[v]]), max(aq[[v]]), length.out = 1000)
  base <- data.frame(Temp     = median(aq$Temp),
    Wind     = median(aq$Wind), Solar.R = median(aq$Solar.R)
  )[rep(1,1000),]
  base[[v]] <- grid
  ## kernel prediction and conf intervals
  outNP  <- predict(npMod, exdat = base[predVARS],
                     se.fit = TRUE)
  meanNP <- outNP$fit
  lowNP  <- meanNP-1.96*outNP$se.fit
  highNP <- meanNP+1.96*outNP$se.fit
  ## knn bootstrap and estimated error bars
  scaledBASE <- scale(base[predVARS], center = scaleTrain$mu,
                      scale  = scaleTrain$sd)
  bootMAT <- matrix(NA, nrow = B, ncol = length(grid))
  set.seed(112025)
  for (b in 1:B) {
    bIDX <- sample(seq_along(yTrain), replace = TRUE)
    Xb    <- xTrain[bIDX, ]
    yb    <- yTrain[bIDX]
    bootMAT[b, ] <- knn.reg(train = Xb, test = scaledBASE,
                            y = yb, k = bestK)$pred
  }
  # .... next page
```



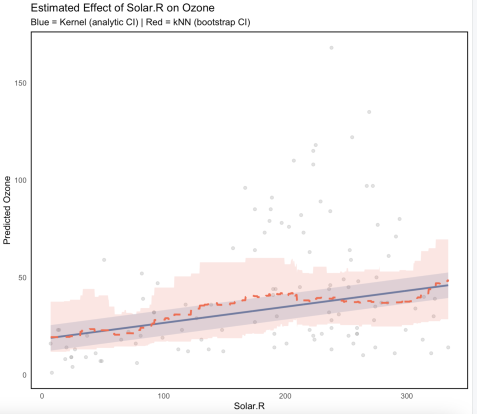
Effect Curves with Confidence Bands

Bootstrap for kNN

```
  meanKNN <- colMeans(bootMAT)
  lowKNN  <- apply(bootMAT, 2, quantile, .025)
  highKNN <- apply(bootMAT, 2, quantile, .975)
p <- ggplot() +
    geom_point(data = aq, aes_string(x=v, y="Ozone"),
               alpha=.3, colour="grey60") +
    geom_ribbon(aes(x=grid, ymin=lowNP, ymax=highNP),
                fill="steelblue", alpha=.22) +
    geom_line(aes(x=grid, y=meanNP),
              colour="steelblue", size=1.05) +
    geom_ribbon(aes(x=grid, ymin=lowKNN, ymax=highKNN),
                fill="tomato",  alpha=.18) +
    geom_line(aes(x=grid, y=meanKNN),
              colour="tomato", size=1.05, linetype="dashed") +
    theme_minimal() +
    labs(
      title = paste0("Estimated Effect of ", v, " on Ozone"),
      subtitle = "Blue = Kernel (analytic CI) | Red = kNN (bootstrap CI)",
      x = v, y = "Predicted Ozone"
    )+
    theme(panel.border = element_rect(NA, "black", 1),
          panel.grid = element_blank())
  print(p)
}
```



Effect Curves with Confidence Bands



Comparing to GAMs (Semi-Parametric)

Generalized Additive Models

Recall the GAM Model

$$\text{Ozone}_i = \beta_0 + s(\text{Wind}_i) + s(\text{Temp}_i) + s(\text{Solar.R}_i) + \beta_4 \text{Month}_i + \epsilon_i$$

Where $s(\cdot)$ are smooth functions estimated with penalized splines.

Final Model Comparison

All models together

```r
###### FINAL SUMMARY ########
## Model comparison table: LM, GAM, kNN, Kernel (npreg)
gamMod <- gam(
  Ozone ~ s(Wind)+s(Temp)+s(Solar.R)+factor(Month),
  data   = train,
  method = "REML")

metrics <- function(y, yhat) {
  rmse <- sqrt(mean((y-yhat)^2))
  mae  <- mean(abs(y-yhat))
  r2   <- 1-sum((y-yhat)^2)/sum((y-mean(y))^2)
  cor_ <- suppressWarnings(cor(y, yhat))
  c(RMSE = rmse, MAE  = mae, ME   = me,
    R2   = r2, COR  = cor_)
}

modelCOMPARE <- modelCOMPARE %>%
  mutate(
    RMSE = round(RMSE, 2),
    MAE  = round(MAE,  2),
    ME   = round(ME,   2),
    R2   = round(R2,   3),
    COR  = round(COR,  3)
  )
```

	Model	Dataset	RMSE	MAE	R2	COR
1	LM	Train	17.07	13.16	0.670	0.818
2	LM	Test	26.34	18.20	0.550	0.761
3	GAM	Train	13.86	10.33	0.782	0.885
4	GAM	Test	23.06	14.92	0.655	0.829
5	kNN (k=4)	Train	13.62	10.04	0.790	0.892
6	kNN (k=4)	Test	22.40	14.67	0.674	0.834
7	Kernel (npreg)	Train	10.86	7.12	0.866	0.932
8	Kernel (npreg)	Test	21.92	15.61	0.688	0.835

Key Takeaways: Nonparametric Methods

When to use what

Linear Regression:

- When relationships are truly linear, need interpretability and inference, or with a limited sample size

k-Nearest Neighbours:

- Local patterns matter more than global, sufficient dense data coverage
- Can handle complex decision boundaries

Kernel Regression:

- Need smooth predictions, want gradients/derivatives
- Do not have assumption of data distributions

GAMs:

- Need additive decomposition of effects
- Want statistical inference on smooth terms

What's Next?

Additional Questions?
Book an Appointment!

End of this Workshop Series

Stay tuned for our advanced R series next term (Clustering and Classification)!

Thank You

Questions?

Workshop Materials:

`https://github.com/csc-ubc-okanagan/ubco-csc-modeling-workshop`

Contact:

jesse.ghashti@ubc.ca