Notes 6: Regression Diagnostics (Part B)

Nathaniel E. Helwig

Department of Statistics
University of Illinois at Urbana-Champaign

Stat 420: Methods of Applied Statistics
Section N1U/N1G – Spring 2014

Outline of Notes

1) Multicollinearity Issues:
- Definition of problem
- Part and partial correlations
- Variance inflation factors

2) Data Transformations:
- Variance stabilizing
- Natural logarithm
- Box-Cox

3) Miscellaneous Topics:
- Standardized regression
- Ridge regression
- Nonparametric regression

4) Linearity Assumption:
- Visualizing non-linearity
- Testing for non-linearity
- Solutions for non-linearity

Multicollinearity: Overview

Consider the MLR model $y_i = b_0 + \sum_{j=1}^{p} b_j x_{ij} + e_i$ with $e_i \overset{\text{iid}}{\sim} \mathrm{N}(0, \sigma^2)$; if the x_{ij} are highly correlated with one another, we have *multicollinearity*.

This is a problem because interpretation becomes difficult...

- b_j is expected change in Y holding other predictors constant
- If predictors are highly correlated, how do we interpret b_j??

Multicollinearity is also a problem for model estimation...

- If predictors are highly correlated, the inverse $(\mathbf{X}'\mathbf{X})^{-1}$ is unstable
- Can not trust the resulting parameter and SE estimates

Part Correlation: Definition

A simple way to quantify the unique contribution of a predictor X_j is the difference in R^2 with and without X_j in the model.

Let $\mathcal{X}$ the denote the space spanned by $\{1, X_1, \ldots, X_p\}$, let $\mathcal{X}_j$ denote the space spanned by X_j for $j \in \{1, \ldots, p\}$, and define $\mathcal{D}_j = \mathcal{X} \ominus \mathcal{X}_j$.

- $\mathcal{D}_j$ is space spanned by $\{1, X_1, \ldots, X_{j-1}, X_{j+1}, \ldots, X_p\} = \{1, D_j\}$

A standardized measure is the *part correlation*, which is defined as

$$r_{Y(X_j \cdot D_j)} = \sqrt{R^2_{\mathcal{X}} - R^2_{\mathcal{D}_j}}$$

where $R^2_{\mathcal{X}}$ and $R^2_{\mathcal{D}_j}$ denote the R^2 with and without X_j, respectively.

Part Correlation: Properties

Part correlation is also called the *semipartial correlation*.

Given predictors X_1, X_2 and response Y, the part (or semipartial) correlation of Y and X_1, controlling for X_2, can be written as

$$r_{Y(X_1 \cdot X_2)} = \frac{r_{YX_1} - r_{YX_2} r_{X_1X_2}}{\sqrt{1 - r^2_{X_1X_2}}}$$

Note that $r_{Y(X_1 \cdot X_2)}$ is the correlation between Y and $(X_1 - \hat{X}_1)$, where $\hat{X}_1 = \hat{\gamma}_0 + \hat{\gamma}_1 X_2$ and $(\hat{\gamma}_0, \hat{\gamma}_1)$ are OLS coefficients predicting X_1 from X_2.

Part Correlation: Example in R

```
> set.seed(123)
> x1=runif(100)
> x2=runif(100)
> y=2+3*x1+4*x2+rnorm(100)
> yx1x2mod=lm(y~x1+x2)
> yx1mod=lm(y~x1)
> yx2mod=lm(y~x2)
> x1x2mod=lm(x1~x2)
> x2x1mod=lm(x2~x1)

> # part correlation of y and x1 (controlling for x2)
> sqrt(summary(yx1x2mod)$r.squared-summary(yx2mod)$r.squared)
[1] 0.540294
> cor(x1x2mod$resid,y)
[1] 0.540294

> # part correlation of y and x2 (controlling for x1)
> sqrt(summary(yx1x2mod)$r.squared-summary(yx1mod)$r.squared)
[1] 0.6410288
> cor(x2x1mod$resid,y)
[1] 0.6410288
```

Partial Correlation: Definition

Given predictors X_1, X_2 and response Y, the partial correlation of Y and X_1, controlling for X_2, can be written as

$$r_{YX_1 \cdot X_2} = \frac{r_{YX_1} - r_{YX_2} r_{X_1 X_2}}{\sqrt{1 - r_{YX_2}^2}\sqrt{1 - r_{X_1 X_2}^2}} = \frac{r_{Y(X_1 \cdot X_2)}}{\sqrt{1 - r_{YX_2}^2}}$$

Note that $\sqrt{1 - r_{YX_2}^2} \leq 1$ with equality holding only when $r_{YX_2}^2 = 0$, which implies that

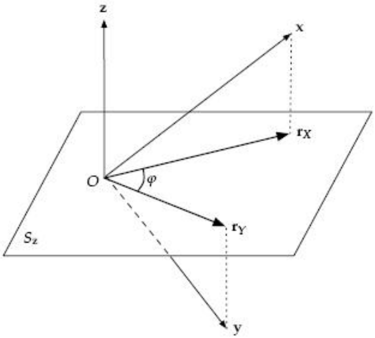
$$r_{YX_1 \cdot X_2} \geq r_{Y(X_1 \cdot X_2)}$$

with equality holding only when $r_{YX_2}^2 = 0$.

Partial Correlation: Properties

Note that $r_{YX_1 \cdot X_2}$ is the correlation between $(Y - \hat{Y}^*)$ and $(X_1 - \hat{X}_1)$, where $\hat{Y}^* = \hat{\kappa}_0 + \hat{\kappa}_1 X_2$ and $\hat{X}_1 = \hat{\gamma}_0 + \hat{\gamma}_1 X_2$.

- $(\hat{\kappa}_0, \hat{\kappa}_1)$ are OLS coefficients predicting Y from X_2.
- $(\hat{\gamma}_0, \hat{\gamma}_1)$ are OLS coefficients predicting X_1 from X_2.



Partial Correlation: R Function

We can define our own part and partial correlation function.

```
pcor=function(x,y,z,type=c("partial","part")){
  rxy=cor(x,y)
  rxz=cor(x,z)
  ryz=cor(y,z)
  pc=(rxy-ryz*rxz)/sqrt(1-rxz^2)
  if(type[1]=="partial"){pc=pc/sqrt(1-ryz^2)}
  pc
}
```

Note: `pcor` calculates partial (or part) correlation between `x` and `y`, controlling for `z`; for part correlation, effect of `z` is removed from `x`.

Partial Correlation: Examples

```
> # part and partial correlation of y and x1 (controlling for x2)
> pcor(x1,y,x2,type="part")
[1] 0.540294
> pcor(x1,y,x2)
[1] 0.6729516
```

```
> # part and partial correlation of y and x2 (controlling for x1)
> pcor(x2,y,x1,type="part")
[1] 0.6410288
> pcor(x2,y,x1)
[1] 0.7335734
```

Variance Inflation Factors: Definition

Another simple way to quantify multicollinearity due to the j-th predictor is the *variance inflation factor*

$$(\mathrm{VIF})_j = \frac{1}{1 - R_j^2}$$

where R_j^2 is the coefficient of multiple determination for predicting X_j from the remaining predictors.

Note that $0 \leq R_j^2 \leq 1$ so we have that...

- $R_j^2 = 0$ implies X_j is linearly unrelated to other predictors
- $R_j^2 = 1$ implies X_j is perfectly linearly related to other predictors
- Larger $(\mathrm{VIF})_j$ values imply more multicollinearity due to X_j

Variance Inflation Factors: R Function

We can use the `vif` function in the `faraway` package.

```
> library(faraway)
> gmod=lm(univ_GPA~.,data=gpa)
> X=model.matrix(gmod)[,-1]
> vif(X)
high_GPA math_SAT verb_SAT comp_GPA
3.715035 4.128046 3.512891 2.795634
```

Rule of thumb: $(\text{VIF})_j > 5$ is cause for concern.

Variance Stabilization: Overview

So far we have assumed that $\mathrm{E}(Y) = \mu$ and $\mathrm{V}(Y) = \sigma^2$ where the mean μ and the variance σ^2 are unrelated parameters.

Instead, suppose that $\mathrm{E}(Y) = \mu$ and $\mathrm{V}(Y) = \psi(\mu)$, i.e., the variance of Y is some function $\psi(\cdot)$ of the mean μ.

Idea: we want to find some function $g(Y)$ such that $\mathrm{V}(g(Y)) = c$, where c is some constant that does not depend on μ.

Variance Stabilization: Derivation

Using a first order Taylor Series expansion for $g(y)$ at $y = \mu$ we have

$$g(y) \approx g(\mu) + g'(\mu)(y - \mu)$$
$$\mathrm{V}(g(y)) \approx [g'(\mu)]^2\psi(\mu)$$

where g' denotes the first derivative of g.

We want to find a function g such that $\mathrm{V}(g(y)) \approx [g'(\mu)]^2\psi(\mu) = c$, and a suitable solution is given by

$$g(\mu) \propto \int \frac{1}{\sqrt{\psi(\mu)}} \mathrm{d}\mu$$

where the proportionality constant can be ignored.

Variance Stabilization: Example #1 (Poisson)

Suppose $Y \sim \text{Poi}(\lambda)$, which denotes that Y follows a Poisson distribution with parameter λ.

- Probability mass function (pmf) is defined for $Y \in \{0, 1, 2, 3, \ldots\}$
- Poisson distribution has one parameter ($\lambda \in \mathbb{R}^+$)
- If $Y \sim \text{Poi}(\lambda)$, then $\mathrm{E}(Y) = \mathrm{V}(Y) = \lambda$

In this case, the variance stabilizing transformation (VST) is

$$g(\lambda) \propto \int \frac{1}{\sqrt{\lambda}} \mathrm{d}\lambda = 2\sqrt{\lambda}$$

so $g(Y) = \sqrt{Y}$ is the VST for a Poisson distributed variable.

Variance Stabilization: Example #2 (Bernoulli)

Suppose $Y \sim \text{Ber}(p)$, which denotes that Y follows a Bernoulli distribution with parameter p.

- Probability mass function (pmf) is defined for $Y \in \{0, 1\}$
- Bernoulli distribution has one parameter: $p = P(Y = 1)$
- If $Y \sim \text{Ber}(p)$, then $\mathrm{E}(Y) = p$ and $\mathrm{V}(Y) = p(1-p)$

In this case, the variance stabilizing transformation (VST) is

$$g(p) \propto \int \frac{1}{\sqrt{p(1-p)}} \mathrm{d}p = 2\sin^{-1}(\sqrt{p})$$

so $g(Y) = \sin^{-1}(\sqrt{Y})$ is the VST for a Bernoulli distributed variable.

Log Transformation: Overview

Suppose that $Y \in \mathbb{R}^+$ and the observed Y values have a large range.

- Y values range over more than one order of magnitude

May be helpful to transform Y using a *logarithm (log) transformation*.

Reminder about logarithms:

- $\log_b(a) = x \iff b^x = a$ where b is the *base* of the logarithm
- Typically bases included $b = 2$ (binary), $b = 10$ (common), and $b = e \approx 2.7183$ (natural)
- In statistics write either $\ln(Y)$ or $\log(Y)$ to denote natural logarithm

Log Transformation: R Function

You can use the `log` function in R for natural logarithm.

```
> # natural logarithm (default)
> set.seed(123)
> x=runif(5)*2
> x
[1] 0.5751550 1.5766103 0.8179538 1.7660348 1.8809346
> lx=log(x)
> lx
[1] -0.5531156  0.4552771 -0.2009494  0.5687368  0.6317688
> exp(1)^lx
[1] 0.5751550 1.5766103 0.8179538 1.7660348 1.8809346
```

Log Transformation: R Function (continued)

Or you can use the `log` function in R for any logarithm.

```
> # common logarithm (base=10)
> set.seed(123)
> x=runif(5)*2
> x
[1] 0.5751550 1.5766103 0.8179538 1.7660348 1.8809346
> lx=log(x,base=10)
> lx
[1] -0.2402151  0.1977244 -0.0872712  0.2469993  0.2743737
> 10^lx
[1] 0.5751550 1.5766103 0.8179538 1.7660348 1.8809346
```

Log Transformation: Regression Interpretation

Consider the MLR model with a log-transformed response

$$\log(y_i) = b_0 + \sum_{j=1}^{p} b_j x_{ij} + e_i$$

with $e_i \overset{\text{iid}}{\sim} \mathrm{N}(0, \sigma^2)$; note that Y has a log-normal distribution.

Linear model on log-scale implies nonlinear model on data scale:

$$\begin{aligned} \hat{y}_i &= e^{\hat{b}_0 + \sum_{j=1}^{p} \hat{b}_j x_{ij}} \\ &= e^{\hat{b}_0} e^{\hat{b}_1 x_{i1}} \cdots e^{\hat{b}_p x_{ip}} \end{aligned}$$

so $e^{\hat{b}_j}$ gives expected multiplicative change in Y corresponding to a 1-unit increase in X_j, holding the other predictors constant.

Box-Cox Transformation: Overview

Consider the MLR model with a transformation of the response

$$g_\lambda(y_i) = b_0 + \sum_{j=1}^{p} b_j x_{ij} + e_i$$

with $e_i \overset{\text{iid}}{\sim} \mathrm{N}(0, \sigma^2)$; note that $g_\lambda(y_i)$ has a normal distribution.

Box and Cox (1964) proposed a method for finding the appropriate transformation of Y in such a situation:

$$g_\lambda(y) = \begin{cases} \frac{y^\lambda - 1}{\lambda} & \lambda \neq 0 \\ \ln(y) & \lambda = 0 \end{cases}$$

which is a power transformation of Y.

Box-Cox Transformation: Estimation

We want to choose λ to maximize the profile log-likelihood

$$L(\lambda) = -\frac{n}{2}\ln(\tilde{\sigma}_\lambda^2) + (\lambda - 1)\sum_{i=1}^n \ln(y_i)$$

where $\tilde{\sigma}_\lambda^2$ is the MLE of the error variance with $g_\lambda(y_i)$ as the response.

Can use numerical approach to maximize $L(\lambda)$, and a $100(1-\alpha)\%$ CI for the optimal power transformation is given by all λ satisfying

$$L(\lambda) > L(\hat{\lambda}) - \frac{1}{2}\chi^2_{1(\alpha)}$$

where $\chi^2_{1(\alpha)}$ is the χ^2_1 quantile with α in the right tail.

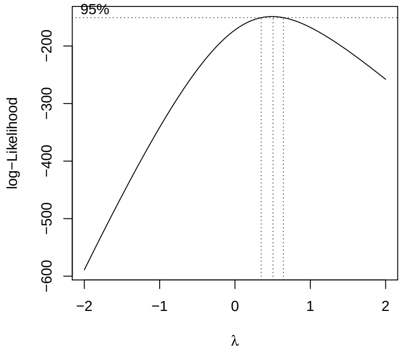
Box-Cox Transformation: R Function

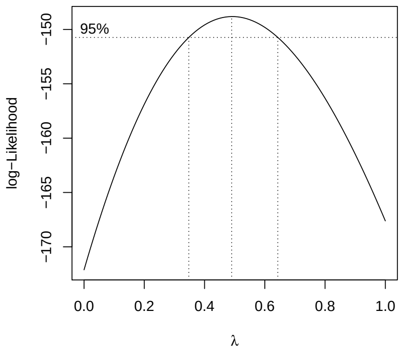
Can use the `boxcox` function in the `MASS` package.

```
> library(MASS)
> par(mfrow=c(1,2))
> set.seed(1234)
> x=runif(100)*2
> y=(2+3*x+rnorm(100))^2
> mymod=lm(y~x)
> boxcox(mymod,plotit=TRUE)
> boxcox(mymod,plotit=TRUE,lambda=seq(0,1,by=.01))
```

Box-Cox Transformation: Example

The R code on the previous slide produced the following two figures:

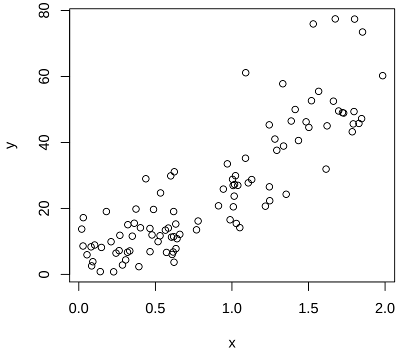


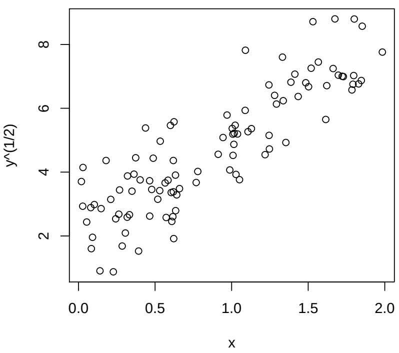


Clearly we can see that the optimal λ is in the interval $\hat{\lambda} \in [0.4, 0.6]$.

Box-Cox Transformation: Example (continued)

Plotting transformed data reveals the transformation was successful:





Standardized Regression: Overview

In MLR each predictor X_j can have a unique scale, so a 1-unit increase in X_1 is not necessarily comparable to a 1-unit increase in X_2.

To make the b_j coefficients more comparable, standardized regression first transforms Y and each X_j to have a mean of 0 and a variance of 1.

Given standardized y_i^* and x_{ij}^*, the MLR model becomes:

$$y_i^* = \sum_{j=1}^{p} b_j^* x_{ij}^* + e_i^*$$

where b_j^* is j-th standardized regression slope.

- b_j^* is expected # of SDs Y will change given 1 SD increase in X_j

Standardized Regression: Properties

There is no intercept, given that $\bar{y}^* = \bar{x}_1^* = \bar{x}_2^* = \cdots = \bar{x}_p^* = 0$.

The $p \times 1$ standardized regression slope vector $\mathbf{b}^*$ can be defined as

$$\hat{\mathbf{b}}^* = \mathbf{R}_{xx}^{-1}\mathbf{r}_{xy}$$

where

- $\mathbf{R}_{xx} = \frac{1}{n-1}(\mathbf{X}^*)'\mathbf{X}^*$ is the $p \times p$ correlation matrix for the columns of the $n \times p$ design matrix $\mathbf{X}^* = (\mathbf{x}_1^*, \ldots, \mathbf{x}_p^*)$
- $\mathbf{r}_{xy} = \frac{1}{n-1}(\mathbf{X}^*)'\mathbf{y}^*$ is the $p \times 1$ correlation vector between the columns of $\mathbf{X}^*$ and the standardized response vector $\mathbf{y}^*$

Standardized Regression: Properties (continued)

Given the OLS solution $\hat{\mathbf{b}} = (\hat{b}_0, \hat{b}_1, \ldots, \hat{b}_p)'$ for the raw (i.e., non-standardized) data, the standardized regression coefficients are

$$\hat{\mathbf{b}}^* = \mathbf{D}\hat{\mathbf{b}}_{-0}$$

where $\hat{\mathbf{b}}_{-0} = (\hat{b}_1, \ldots, \hat{b}_p)'$ and $\mathbf{D} = \mathrm{diag}(s_{x1}/s_y, \ldots, s_{xp}/s_y)$ with s_{xj} and s_y denoting the sample standard deviations of x_{ij} and y_i.

Similarly, given $\hat{\mathbf{b}}^*$ for the standardized data, we have that

$$\hat{\mathbf{b}}_{-0} = \mathbf{D}^{-1}\hat{\mathbf{b}}^*$$

and $\hat{\mathbf{b}} = (\hat{b}_0, \hat{\mathbf{b}}'_{-0})'$ where $\hat{b}_0 = \bar{y} - \sum_{j=1}^{p} \hat{b}_j \bar{x}_j$.

Standardized Regression: R Function

We can design an R function to standardize a result output from `lm`:

```
sdlm=function(mymod){
  xsds=apply(mymod$model[,-1],2,sd)
  ysds=sd(mymod$model[,1])
  mymod$coef[-1]*(xsds/ysds)
}
```

Note that the `-1` index excludes the first element (column or entry).

Standardized Regression: Example

```
> # generate multiple regression data
> set.seed(123)
> x1=runif(100)
> x2=runif(100)
> X=cbind(x1,x2)
> y=2+3*x1+4*x2+rnorm(100)
> mymod=lm(y~x1+x2)
> mymod$coef
(Intercept)          x1          x2
   1.859206    3.097260    3.970358
```

Standardized Regression: Example (continued)

```
> # scale data to have mean 0 and variance 1
> ys=scale(y)
> x1s=scale(x1)
> x2s=scale(x2)
> Xs=cbind(x1s,x2s)
```

```
> # solve for standardized coefficients
> bstar=solve(cor(Xs))%*%cor(Xs,ys)
> bstar
          [,1]
[1,] 0.5423633
[2,] 0.6434840
```

Standardized Regression: Example (continued)

```
> # compare to our sdlm function
> sdlm(mymod)
       x1        x2
0.5423633 0.6434840

> # compare to solution with ys, x1s, and x2s
> lm(ys~x1s+x2s)$coef
  (Intercept)           x1s           x2s
-1.599246e-17  5.423633e-01  6.434840e-01
```

Ridge Regression: Overview

Ridge regression penalizes squared magnitude of the **b** coefficients.

- Introduces bias (shrinkage) to the coefficient estimates
- Reduces the variance of the coefficient estimates

Ridge regression finds $\hat{\mathbf{b}}_\lambda$ to minimize penalized least-squares function

$$\|\mathbf{y} - \mathbf{X}\mathbf{b}\|^2 + \lambda\|\mathbf{b}\|^2$$

where $\lambda \geq 0$ is the ridge parameter such that

- As $\lambda \to 0$, $\hat{\mathbf{b}}_\lambda \to \hat{\mathbf{b}}$ (i.e., ridge estimate approaches OLS estimate)
- As $\lambda \to \infty$, $\hat{\mathbf{b}}_\lambda \to \mathbf{0}$ (i.e., ridge estimate approaches zero vector)

Ridge Regression: Estimation

Given the ridge parameter λ, the coefficient estimates are given by

$$\hat{\mathbf{b}}_\lambda = (\mathbf{X}'\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}'\mathbf{y}$$

where $\mathbf{I}$ denotes the identity matrix of order $p+1$.

Corresponding fitted values are given by

$$\hat{\mathbf{y}}_\lambda = \mathbf{X}\hat{\mathbf{b}}_\lambda = \mathbf{H}_\lambda\mathbf{y}$$

where $\mathbf{H}_\lambda = \mathbf{X}(\mathbf{X}'\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}'$ is ridge hat matrix, which depends on λ.

Ridge Regression: Estimation (continued)

Pick the optimal λ by minimizing a cross-validation statistic.

Craven and Wabha'a (1979) Generalized Cross-Validation (GCV) score is given by

$$\mathrm{GCV}(\lambda) = \frac{n \sum_{i=1}^{n} (y_i - \hat{y}_{i\lambda})^2}{[n - \mathrm{tr}(\mathbf{H}_\lambda)]^2}$$

where $\mathrm{tr}(\mathbf{H}_\lambda)$ is the *effective degrees of freedom* of the fit model.

Ridge Regression: R Function

You can use the `lm.ridge` function in the `MASS` package.

Default use of `lm.ridge` produces OLS estimates ($\lambda = 0$).

Enter a vector of (positive) λ's to try different ridge parameters.

Ridge Regression: Example

```
> # generate data with multicollinearity
> library(MASS)
> set.seed(123)
> x1=scale(rnorm(100))
> x2=scale(x1+rnorm(100,sd=.2))
> cor(x1,x2)
          [,1]
[1,] 0.9815079
> y=3*x1+4*x2+scale(rnorm(100))
> mymod=lm(y~0+x1+x2)
> summary(mymod)$coef
   Estimate Std. Error  t value     Pr(>|t|)
x1 2.746686  0.5231333 5.250452 8.816854e-07
x2 4.126477  0.5231333 7.888003 4.394362e-12
> select(lm.ridge(y~0+x1+x2,lambda=seq(0,0.1,.0001)))
modified HKB estimator is 0
modified L-W estimator is 0
smallest value of GCV  at 0.1
> lm.ridge(y~0+x1+x2,lambda=0.1)
      x1       x2
2.780346 4.089350
```

Ridge Regression: Example (continued)

```
> X=cbind(x1,x2)
> XtX=crossprod(X)
> W=solve(XtX+0.1*diag(2))
> bhat=tcrossprod(W,X)%*%y
> H=X%*%tcrossprod(W,X)
> yhat=H%*%y
> edf=sum(diag(H))
> sigmasq=sum((y-yhat)^2)/(100-edf)
> bcov=sigmasq*W%*%XtX%*%W
> bsev=sqrt(diag(bcov))
> mytab=cbind(bhat,bsev,bhat/bsev,2*(1-pt(abs(bhat/bsev),100-edf)))
> colnames(mytab)=c("Estimate","Std. Error", "t value", "Pr(>|t|))")
> rownames(mytab)=c("x1","x2")
> mytab
   Estimate Std. Error  t value   Pr(>|t|))
x1 2.780668  0.4961753 5.604204 1.921941e-07
x2 4.088993  0.4961753 8.241025 7.718270e-13
```

Nonparametric Regression: Overview

So far we have assumed the relationship between X and Y is linear; nonparametric regression extends this to nonlinear functions

$$y_i = \eta(x_i) + e_i$$

where η is some smooth function of $x_i \in \mathcal{X}$ and $e_i \overset{\mathrm{iid}}{\sim} \mathrm{N}(0, \sigma^2)$.

Using cubic smoothing splines, we can write the smooth function as

$$\eta(x) = \eta_0 + \eta_{\mathrm{n}}(x) + \eta_{\mathrm{c}}(x)$$

where η_0 is a constant function, η_{n} is a null space (linear) function, and η_{c} is a contrast space (nonlinear) function.

Nonparametric Regression: Representation

Find $\hat{\eta}_\lambda$ to minimize the penalized least squares functional

$$\sum_{i=1}^{n}(y_i - \eta(x_i))^2 + \lambda n J(\eta)$$

where $J(\cdot)$ is a nonnegative penalty functional quantifying the roughness of η and $\lambda \geq 0$ is the smoothing parameter.

- Cubic spline with $x \in [0, 1]$: $J(\eta) = \int_0^1 [\ddot{\eta}(x)]^2 dx$
- $\ddot{\eta}$ is the second derivative of η w.r.t. x

To represent the function, we use the reproducing kernels (RKs) of $\mathcal{X}$

$$\eta(x) = \sum_{v=1}^{m} d_v \phi_v(x) + \sum_{u=1}^{q} c_u \rho_{\mathrm{c}}(x, x_u^*)$$

where $\{\phi_v\}_{v=1}^{m}$ span the null space, ρ_{c} is the RK of the contrast space, and $\{x_u^*\}_{u=1}^{q}$ are the selected *knots* for the spline such that $x_u^* \in \mathcal{X}$.

Nonparametric Regression: Estimation

Using this representation for η, we can rewrite the penalized least-squares functional as

$$\|\mathbf{y} - \mathbf{Kd} - \mathbf{Jc}\|^2 + \lambda n \mathbf{c}'\mathbf{Qc}$$

where

- $\mathbf{y} = (y_1, \ldots, y_n)'$ is response vector
- $\mathbf{K} = \{\phi_v(x_i)\}_{n \times m}$ is null space basis function matrix
- $\mathbf{J} = \{\rho_{\mathrm{c}}(x_i, x_u^*)\}_{n \times q}$ is contrast space basis function matrix
- $\mathbf{Q} = \{\rho_{\mathrm{c}}(x_t^*, x_u^*)\}_{q \times q}$ is penalty matrix
- $\mathbf{d} = (d_1, \ldots, d_m)'$ and $\mathbf{c} = (c_1, \ldots, c_q)'$ are unknown coefficients

Note: $\mathbf{c}'\mathbf{Qc} = J(\eta)$ due to the properties of RKs!

Nonparametric Regression: Estimation (continued)

The coefficients minimizing the penalized least-squares function are

$$\begin{pmatrix} \hat{\mathbf{d}} \\ \hat{\mathbf{c}} \end{pmatrix} = \begin{pmatrix} \mathbf{K}'\mathbf{K} & \mathbf{K}'\mathbf{J} \\ \mathbf{J}'\mathbf{K} & \mathbf{J}'\mathbf{J} + \lambda n \mathbf{Q} \end{pmatrix}^{\dagger} \begin{pmatrix} \mathbf{K}' \\ \mathbf{J}' \end{pmatrix} \mathbf{y} \tag{1}$$

where $(\cdot)^{\dagger}$ denotes the Moore-Penrose pseudoinverse.

The fitted values are given by $\hat{\mathbf{y}} = \mathbf{K}\hat{\mathbf{d}} + \mathbf{J}\hat{\mathbf{c}} = \mathbf{S}_{\lambda}\mathbf{y}$ where

$$\mathbf{S}_{\lambda} = \begin{pmatrix} \mathbf{K} & \mathbf{J} \end{pmatrix} \begin{pmatrix} \mathbf{K}'\mathbf{K} & \mathbf{K}'\mathbf{J} \\ \mathbf{J}'\mathbf{K} & \mathbf{J}'\mathbf{J} + \lambda n \mathbf{Q} \end{pmatrix}^{\dagger} \begin{pmatrix} \mathbf{K}' \\ \mathbf{J}' \end{pmatrix} \tag{2}$$

is the *smoothing matrix*, which depends on λ.

Nonparametric Regression: R Functions

There are a few R functions for nonparametric regression:

- `ssanova` function in `gss` package
- `gam` function in `mgcv` package
- `ssr` function in `assist` package

In about two weeks, you can use my `bigsplines` package.

- `bigspline` fits unidimensional splines
- `bigssa` fits n-way SSAs (nonparametric regression)
- `bigssp` fits semiparametric regression

Nonparametric Regression: Example

```
> # generate nonlinear data
> library(gss)
> set.seed(123)
> x=runif(100)
> y=sin(2*pi*x)+rnorm(100)
```

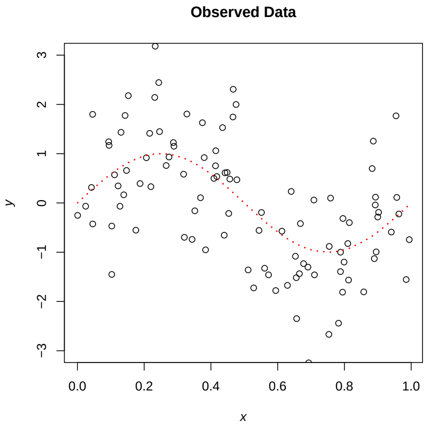
```
> # fit cubic spline with q=20 knots
> mymod=ssanova(y~x,type="cubic",nbasis=20)
```

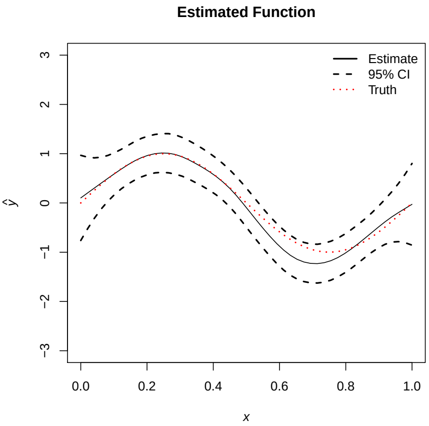
```
> # predict and get standard errors
> newdata=data.frame(x=seq(0,1,l=50))
> yhat=predict(mymod,newdata,se.fit=TRUE)
```

Nonparametric Regression: Example (continued)

```
> # plot results with 95% CI
> x11(width=12,height=6)
> par(mfrow=c(1,2),mar=c(5,4.5,4,2)+0.1)
> plot(x,y,xlab=expression(italic(x)),ylab=expression(italic(y)),
+      ylim=c(-3,3),main="Observed Data")
> lines(sort(x),sin(2*pi*sort(x)),lty=3,col="red",lwd=2)
> plot(newdata$x,yhat$fit,xlab=expression(italic(x)),
+      ylab=expression(italic(hat(y))),type="l",ylim=c(-3,3),
+      main="Estimated Function")
> lines(newdata$x,yhat$fit+qnorm(.975)*yhat$se.fit,lty=2,lwd=2)
> lines(newdata$x,yhat$fit-qnorm(.975)*yhat$se.fit,lty=2,lwd=2)
> lines(newdata$x,sin(2*pi*newdata$x),lty=3,col="red",lwd=2)
> legend("topright",c("Estimate","95% CI","Truth"),lty=1:3,
+        lwd=rep(2,3),col=c(rep("black",2),"red"),bty="n")
```

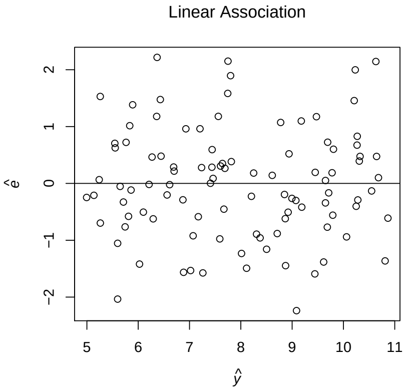
Nonparametric Regression: Example (continued)

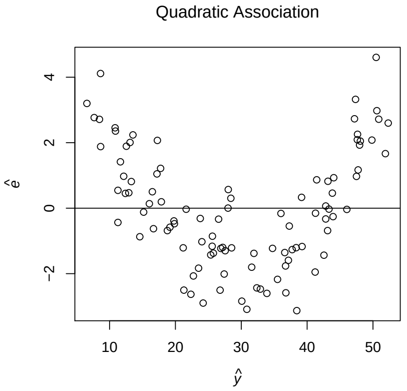




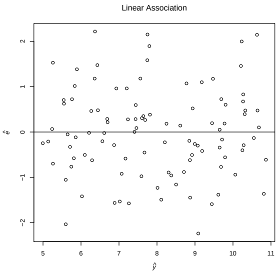
Residual Plots

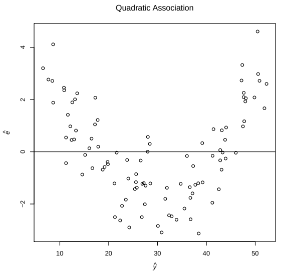
To visualize the linearity assumption, plot $\hat{y}_i$ versus $\hat{e}_i$:

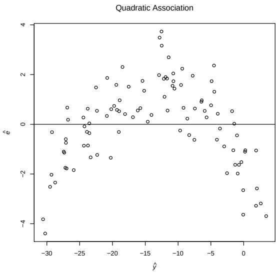


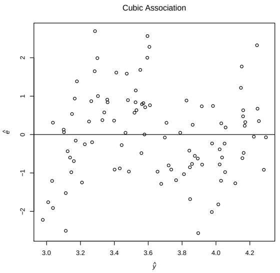


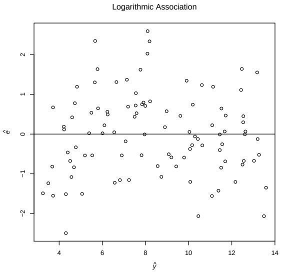
Different Residual Trends

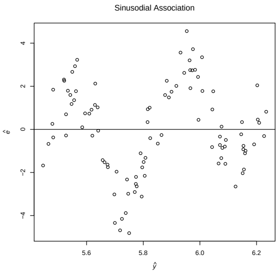












Testing for Non-Linearity: Overview

To test for a significant non-linear association you could:

- Find the optimal Box-Cox transformation
- Fit a nonparametric regression model

We will focus on nonparametric regression because it is more general.

Testing for Non-Linearity: Example #1 (Linear)

```
> # generate linear data
> library(gss)
> set.seed(123)
> x=runif(100)
> y=2+4*x+rnorm(100)
```

```
> # fit cubic spline with q=20 knots
> mymod=ssanova(y~x,type="cubic",nbasis=20)
> summary(mymod)$r.squared
[1] 0.5717828
> summary(lm(y~x))$r.squared
[1] 0.5717828
```

Testing for Non-Linearity: Example #2 (Non-Linear)

```
> # generate nonlinear data
> library(gss)
> set.seed(123)
> x=runif(100)
> y=sin(2*pi*x)+rnorm(100)
```

```
> # fit cubic spline with q=20 knots
> mymod=ssanova(y~x,type="cubic",nbasis=20)
> summary(mymod)$r.squared
[1] 0.3883189
> summary(lm(y~x))$r.squared
[1] 0.2290272
```

Solutions for Non-Linearity

If you have data with a nonlinear relationship, you could

- Transform data to have more linear relationship (e.g., Box-Cox)
- Fit polynomial regression model
- Fit nonparametric regression model
- Use other nonparametric approach (e.g., analyze rank data)