Elements of Nonparametric Statistics

Nicholas Henderson

2020-04-20

Contents

Preface

This book will serve as the main source of course notes for Biostatistics 685/Statistics 560, Winter 2020.

Chapter 1

Introduction

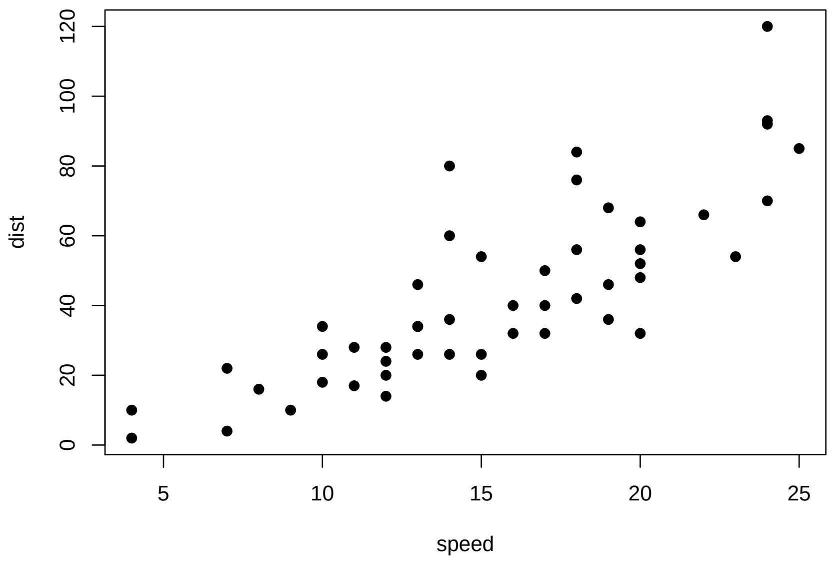


Figure 1.1: Hello World!

1.1 What is Nonparametric Statistics?

What is Parametric Statistics?

- Parametric models refer to probability distributions that can be fully described by a fixed number of parameters that do not change with the sample size.
- Typical examples include
  - Gaussian
  - Poisson
  - Exponential
  - Beta
- Could also refer to a regression setting where the mean function is described by a fixed number of parameters.

What is Nonparametric Statistics?

- It is difficult to give a concise, all-encompassing definition, but nonparametric statistics generally refers to statistical methods where there is not a clear parametric component.
- A more practical definition is that nonparametric statistics refers to flexible statistical procedures where very few assumptions are made regarding the distribution of the data or the form of a regression model.
- The uses of nonparametric methods in several common statistical contexts are described in Sections 1.3 - 1.7.

1.2 Outline of Course

This course is roughly divided into the following 5 categories.

1. **Nonparametric Testing**

- Rank-based Tests
- Permutation Tests

1. **Estimation of Basic Nonparametric Quantities**

- The Empirical Distribution Function
- Density Estimation

1. **Nonparametric Confidence Intervals**

- Bootstrap

- Jacknife

1. **Nonparametric Regression Part I (Smoothing Methods)**

- Kernel Methods
- Splines
- Local Regression

1. **Nonparametric Regression Part II (Machine Learning Methods)**

- Decision Trees/CART
- Ensemble Methods

1.3 Example 1: Nonparametric vs. Parametric Two-Sample Testing

Suppose we have data from two groups. For example, outcomes from two different treatments.

- **Group 1 outcomes**: $X_1, \ldots, X_n$ an i.i.d (independent and identically distributed) sample from distribution function F_X. This means that
$$F_X(t) = P(X_i \leq t) \quad \text{for any } 1 \leq i \leq n$$
- **Group 2 outcomes**: $Y_1, \ldots, Y_m$ an i.i.d. sample from distribution function F_Y.
$$F_Y(t) = P(Y_i \leq t) \quad \text{for any } 1 \leq i \leq n$$
- To test the impact of a new treatment, we usually want to test whether or not F_X differs from F_Y in some way. This can be stated in hypothesis testing language as
$$\begin{aligned} H_0 &: F_X = F_Y \quad (\text{ populations are the same}) \\ H_A &: F_X \neq F_Y \quad (\text{ populations are different}) \end{aligned} \tag{1.1}$$

Parametric Tests

- Perhaps the most common parametric test for (1.1) is the **t-test**. The t-test assumes that
$$F_X = \text{Normal}(\mu_x, \sigma^2) \quad \text{and} \quad F_Y = \text{Normal}(\mu_y, \sigma^2) \tag{1.2}$$

- Under this parametric assumption, the hypothesis test (1.1) reduces to

$$H_0 : \mu_x = \mu_y \quad \text{vs.} \quad H_A : \mu_x \neq \mu_y \tag{1.3}$$

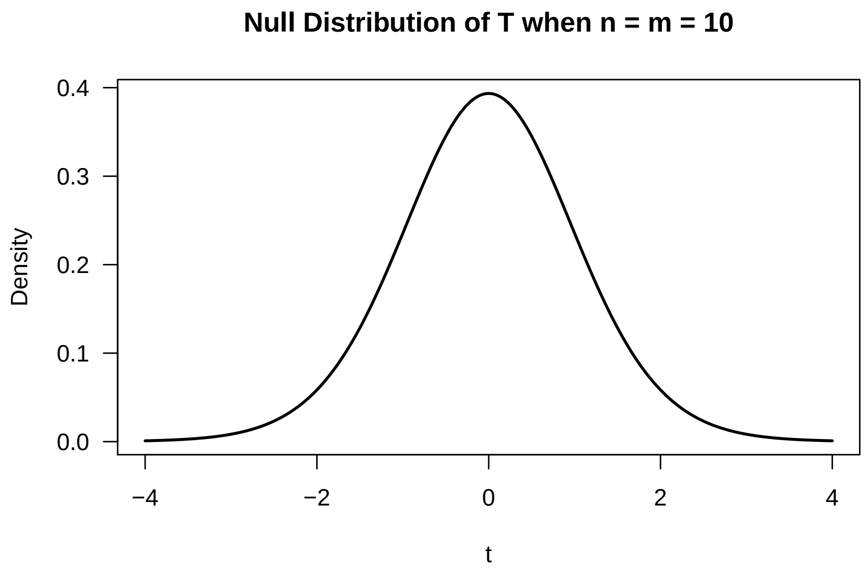
- The standard t-statistic (with a pooled estimate of σ^2) is the following

$$T = \frac{\bar{X} - \bar{Y}}{s_p\sqrt{\frac{1}{n} + \frac{1}{m}}}, \tag{1.4}$$

where $\bar{X} = \frac{1}{n}\sum_{i=1}^{n} X_i$ and $\bar{Y} = \frac{1}{m}\sum_{i=1}^{m} Y_i$ are the group-specific sample means and s_p^2 is the pooled estimate of σ^2

$$s_p^2 = \frac{1}{m+n-2}\Big\{\sum_{i=1}^{n}(X_i - \bar{X})^2 + \sum_{i=1}^{m}(Y_i - \bar{Y})^2\Big\} \tag{1.5}$$

- The t-test is based on the **null distribution** of T - the distribution of T under the null hypothesis.
- Under the assumption of normality, the null distribution of T is a t distribution with $n + m - 2$ degrees of freedom.



- Notice that the null distribution of T depends on the parametric assumption that both $F_X = \text{Normal}(\mu_x, \sigma^2)$ and $F_Y = \text{Normal}(\mu_y, \sigma^2)$. Appealing to the Central Limit Theorem, one could argue that is a quite reasonable assumption.
- In addition to using the assumption that $F_X = \text{Normal}(\mu_x, \sigma^2)$ and $F_Y = \text{Normal}(\mu_y, \sigma^2)$, we used this parametric assumption (at least implicitly) in the formulation of the hypothesis test itself because we assumed that any difference between F_X and F_Y would be fully described by difference in μ_x and μ_y.
- So, in a sense, you are using the assumption of normality twice in the construction of the two-sample t-test.

Nonparametric Tests

- Two-sample nonparametric tests are meant to be "distribution-free". This means the null distribution of the test statistic does not depend on any parametric assumptions about the two populations F_X and F_Y.
- Many such tests are based on **ranks**. The distribution of the ranks under the assumption that $F_X = F_Y$ do not depend on the form of F_X (assuming F_X is continuous).
- Also, the statements of hypotheses tests for nonparametric tests should not rely on any parametric assumptions about F_X and F_Y.
- For example, $H_A : F_X \neq F_Y$ or $H_A : F_X \geq F_Y$.

- Nonparametric tests usually tradeoff power for greater robustness.
- In general, if the parametric assumptions are correct, a nonparametric test will have less power than its parametric counterpart.
- If the parametric assumptions are not correct, parametric tests might have inappropriate type-I error control or lose power.

1.4 Example 2: Nonparametric Estimation

- Suppose we have n observations $(X_1, \dots, X_n)$ which are assumed to be i.i.d. (independent and identically distributed). The distribution function of X_i is F_X.

- Suppose we are interested in estimating the entire distribution function F_X rather than specific features of the distribution of X_i such as the mean or standard deviation.
- In a **parametric** approach to estimating F_X, we would assume the distribution of X_i belongs to some parametric family of distributions. For example,
  - $X_i \sim \text{Normal}(\mu, \sigma^2)$
  - $X_i \sim \text{Exponential}(\lambda)$
  - $X_i \sim \text{Beta}(\alpha, \beta)$

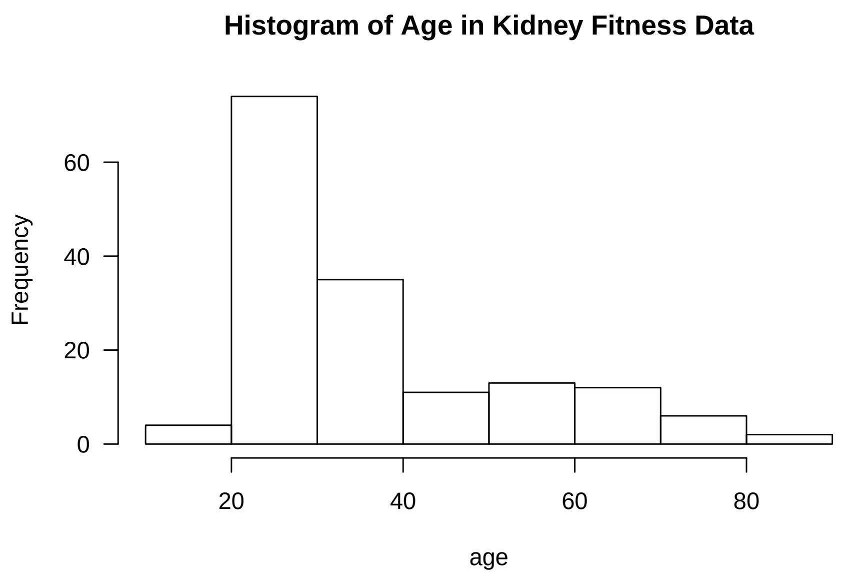
- If we assume that $X_i \sim \text{Normal}(\mu, \sigma^2)$, we only need to estimate 2 parameters to fully describe the distribution of X_i, and the number of parameters will not depend on the sample size.
- In a nonparametric approach to characterizing the distribution of X_i, we need to instead estimate the entire distribution function F_X or density function f_X.
- The distribution function F_X is usually estimated by the **empirical distribution function**

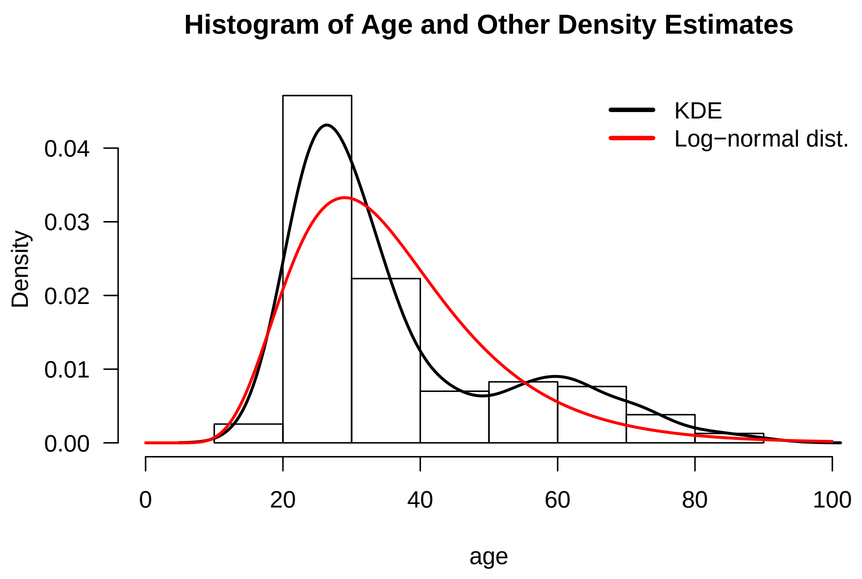
$$\hat{F}_n(t) = \frac{1}{n}\sum_{i=1}^{n} I(X_i \leq t), \tag{1.6}$$

  where $I()$ denotes the indicator function. That is, $I(X_i \leq t) = 1$ if $X_i \leq t$, and $I(X_i \leq t) = 0$ if $X_i > t$.
- The empirical distribution function is a discrete distribution function, and it can be thought of as an estimate having n ”parameters.
- The density function of X_i is often estimated by a kernel density estimator (KDE). This is defined as

$$\hat{f}_n(t) = \frac{1}{nh_n}\sum_{i=1}^{n} K\Big(\frac{t - X_i}{h_n}\Big). \tag{1.7}$$

- $K()$ - the kernel function
- h_n - the bandwidth
- The KDE is a type of smoothing procedure.


Histogram of Age in Kidney Fitness Data
Frequency
0
20
40
60
20
40
60
80
age


Histogram of Age and Other Density Estimates
KDE
Log−normal dist.
Density
0.00
0.01
0.02
0.03
0.04
0
20
40
60
80
100
age

1.5 Example 3: Confidence Intervals

- Inference for a wide range of statistical procedures is based on the following argument

$$\hat{\theta}_n \text{ has an approximate Normal}\Big(\theta, \widehat{\text{Var}(\hat{\theta}_n)}\Big) \text{ distribution} \tag{1.8}$$

- Above, $\hat{\theta}_n$ is an estimate of a parameter θ, and $\widehat{\text{Var}(\hat{\theta}_n)}$ is an estimate of the variance of $\hat{\theta}_n$.
- $se_n = \sqrt{\widehat{\text{Var}(\hat{\theta}_n)}}$ is usually referred to as the **standard error**.
- 95% confidence intervals are reported using the following formula

$$[\hat{\theta}_n - 1.96 se_n, \hat{\theta}_n + 1.96 se_n] \tag{1.9}$$

- Common examples of this include:

  1. $\hat{\theta}_n = \bar{X}_n$.

  In this case, appeals to the Central Limit Theorem would justify approximation (1.8). The variance of $\hat{\theta}_n$ would be σ^2, and the standard error would typically be $se_n = \hat{\sigma}/\sqrt{n}$.

  2. $\hat{\theta}_n =$ Maximum Likelihood Estimate of θ.

  In this case, asymptotics would justify the approximate distribution $\hat{\theta}_n \sim \text{Normal}(\theta, \frac{1}{nI(\theta)})$, where $I(\theta)$ denotes the Fisher information. The standard error in this context is often $se_n = \{nI(\hat{\theta}_n)\}^{-1/2}$.

- Confidence intervals using (1.8) rely on a parametric approximation to the sampling distribution of the statistic $\hat{\theta}_n$.
- Moreover, even if one wanted to use something like (1.8), working out standard error formulas can be a great challenge in more complicated situations.

- The **bootstrap** is a simulation-based approach for computing standard errors and confidence intervals.

- The bootstrap does not rely on any particular parametric assumptions and can be applied in almost any context (though bootstrap confidence intervals can fail to work as desired in some situations).
- Through resampling from the original dataset, the bootstrap uses many possible alternative datasets to assess the variability in $\hat{\theta}_n$.

	OriginalDat	Dat1	Dat2	Dat3	Dat4
Obs. 1	0.20	0.20	0.80	0.20	0.30
Obs. 2	0.50	0.20	0.80	0.20	0.70
Obs. 3	0.30	0.30	0.50	0.80	0.20
Obs. 4	0.80	0.30	0.70	0.50	0.50
Obs. 5	0.70	0.70	0.20	0.30	0.20
theta.hat	0.50	0.34	0.60	0.40	0.38

- In the above example, we have 4 **boostrap replications** for the statistic $\hat{\theta}$:

$$\hat{\theta}^{(1)} = 0.34 \quad (1.10)$$
$$\hat{\theta}^{(2)} = 0.60 \quad (1.11)$$
$$\hat{\theta}^{(3)} = 0.40 \quad (1.12)$$
$$\hat{\theta}^{(4)} = 0.38 \quad (1.13)$$

- In the above example, the bootstrap standard error for $\hat{\theta}_n$ would be the standard deviation of the bootstrap replications

$$\begin{aligned} se_{boot} &= \Big(\frac{1}{3}\sum_{b=1}^{4}\{\hat{\theta}^{(b)} - \hat{\theta}^{(-)}\}^2\Big)^{1/2} \\ &= \Big((0.34-0.43)^2/3 + (0.60-0.43)^2/3 + (0.40-0.43)^2/3 + (0.38-0.43)^2/3\Big)^{1/2} \\ &= 0.116 \qquad (1.14) \end{aligned}$$

where $\hat{\theta}^{(-)} = 0.43$ is the average of the bootstrap replications.

- One would then report the confidence interval $[\hat{\theta} - 1.96 \times 0.116, \hat{\theta} + 1.96 \times 0.116]$. In practice, the number of bootstrap replications is typically much larger than 4.
- It is often better to construct confidence intervals using the percentiles from the bootstrap distribution of $\hat{\theta}$ rather than use a confidence interval of the form: $\hat{\theta} \pm 1.96 \times se_{boot}$.

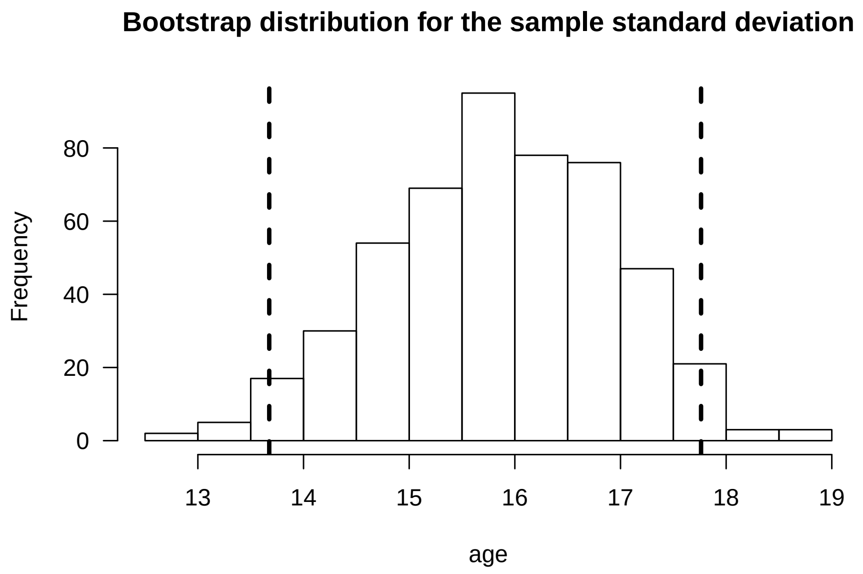
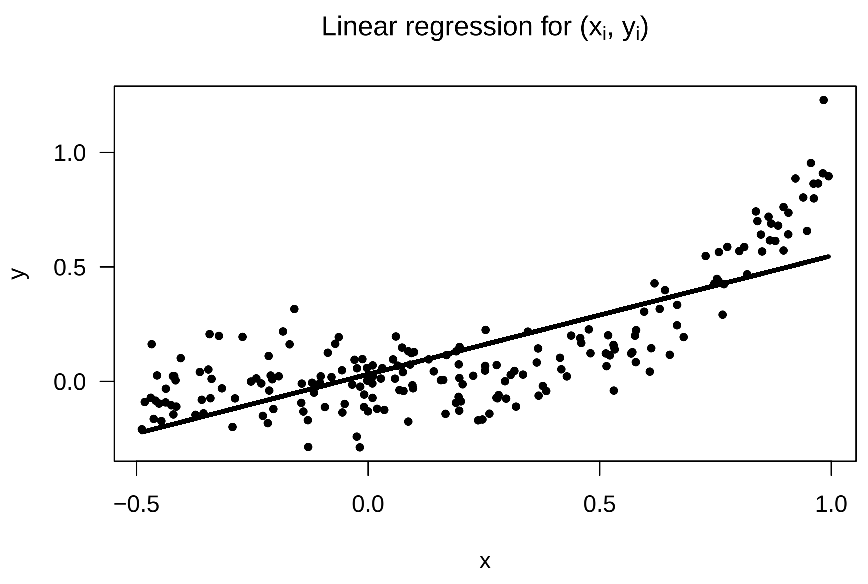


Figure 1.2: Bootstrap distribution of the sample standard deviation for the age variable from the kidney fitness data. Dasjed vertical lines are placed at the 2.5 and 97.5 percentiles of the bootstrap distribution.

1.6 Example 4: Nonparametric Regression with a Single Covariate

- Regression is a common way of modeling the relationship between two different variables.
- Suppose we have n pairs of observations $(y_1, x_1), \dots, (y_n, x_n)$ where y_i and x_i are suspected to have some association.
- Linear regression would assume that these y_i and x_i are related by the following
$$y_i = \beta_0 + \beta_1 x_i + \varepsilon_i \tag{1.15}$$
with the assumption $\varepsilon_i \sim \text{Normal}(0, \sigma^2)$ often made.
- In this model, there are only 3 parameters: $(\beta_0, \beta_1, \sigma^2)$, and the number of parameters stays fixed for all n.



- The nonparametric counterpart to linear regression is usually formulated in the following way
$$y_i = m(x_i) + \varepsilon_i \tag{1.16}$$

- Typically, one makes very few assumptions about the form of the mean function m, and it is not assumed m can be described by a finite number of parameters.

- There are a large number of nonparametric methods for estimating m.

- One popular method is the use of **smoothing splines**.

- With smoothing splines, one considers mean functions of the form

$$m(x) = \sum_{j=1}^{n} \beta_j g_j(x) \tag{1.17}$$

where $g_1, \dots, g_n(x)$ are a collection of spline basis functions.

- Because of the large number of parameters in (1.17), one should estimate the basis function weights β_j through penalized regression

$$\text{minimize} \quad \sum_{i=1}^{n} \Big(y_i - \sum_{j=1}^{n} \beta_j g_j(x_i) \Big)^2 + \lambda \sum_{i=1}^{n} \sum_{j=1}^{n} \Omega_{ij} \beta_i \beta_j \tag{1.18}$$

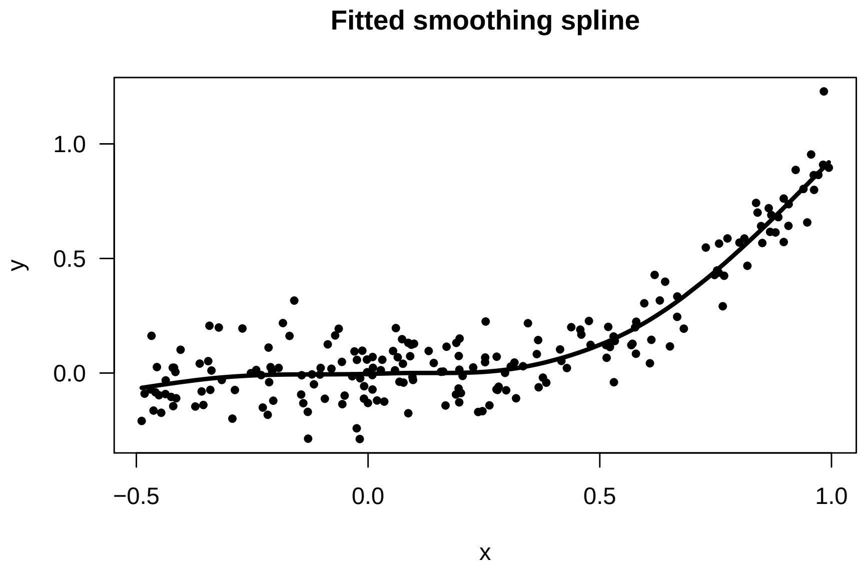
where $\Omega_{ij} = \int g_i''(t) g_j''(t) dt$.

- Using coefficient estimates $\hat{\beta}_1, \dots, \hat{\beta}_n$ found from solving (1.17), the nonparametric estimate of the mean function is defined as

$$\hat{m}(x) = \sum_{j=1}^{n} \hat{\beta}_j g_j(x) \tag{1.19}$$

- While the estimation in (1.18) resembles parametric estimation for linear regression, notice that the number of parameters to be estimated will change with the sample size.

- Allowing the number of basis functions to grow with n is important. For a sufficiently large number of basis functions, one should be able to approximate the true mean function $m(x)$ arbitrarily closely.



1.7 Example 5: Classification and Regression Trees (CART)

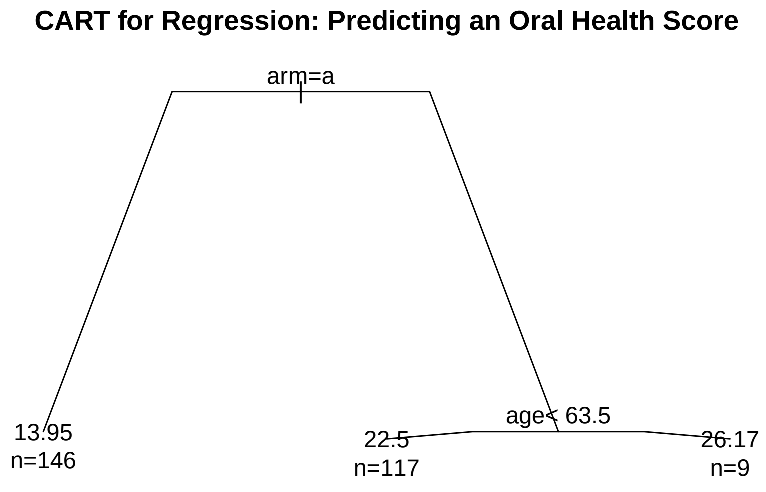
- Suppose we now have observations $(y_1, \mathbf{x}_1), \dots, (y_n, \mathbf{x}_n)$ where y_i is a continuous response and $\mathbf{x}_i$ is a p-dimensional vector of covariates.

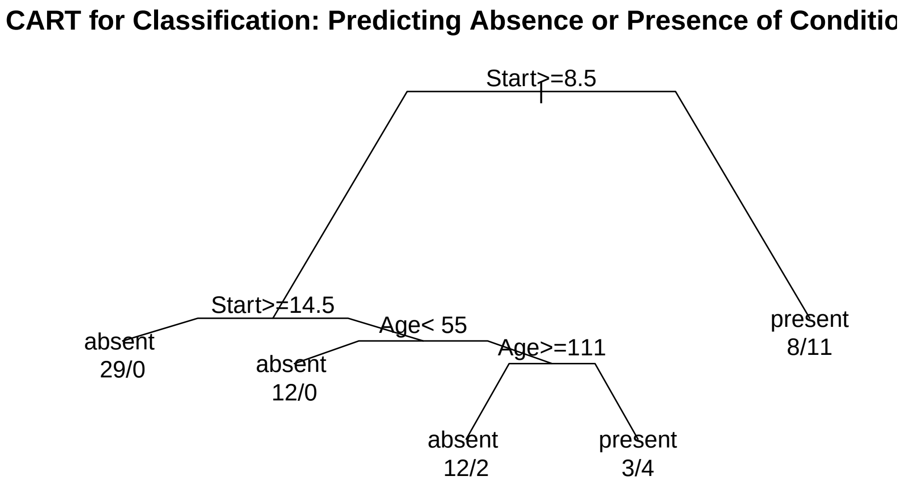
- Regression trees are a nonparametric approach for predicting y_i from $\mathbf{x}_i$.

- Here, the regression function is a **decision tree** rather than some fitted curve.

- With a decision tree, a final prediction from a covariate vector $\mathbf{x}_i$ is obtained by answering a sequence of "yes or no" questions.

- When the responses y_i are binary, such trees are referred to as classification trees. Hence, the name: classification and regression trees (CART).





- Classification and regression trees are constructed through **recursive partitioning**.

- Recursive partitioning is the process of deciding if and how to split a given node into two child nodes.
- Tree splits are usually chosen to minimize the "within-node" sum of squares.
- The size of the final is determined by a process of "pruning" the tree with cross-validation determining the best place to stop pruning.
- Regression trees are an example of a more algorithmic approach to constructing predictions (as opposed to probability modeling in more traditional statistical methods) with a strong emphasis on predictive performance as measured through cross-validation.

- While single regression trees have the advantage of being directly interpretable, their prediction performance is often not that great.
- However, using collections of trees can be very effective for prediction and has been used in many popular learning methods. Examples include: random forests, boosting, and Bayesian additive regression trees (BART).
- Methods such as these can perform well on much larger datasets. We will discuss additional methods if time allows.

Chapter 2

Working with R

You can download R by visiting https://www.r-project.org/ and clicking on the **download R** link. Follow the instructions to complete installation. THe most recent version is version 3.6.2.

It is not necessary to use this, but I find **RStudio** to be a very useful integrated development environment (IDE) for computing with **R**. **RStudio** may be downloaded and installed by visiting https://rstudio.com/

Part I

Nonparametric Testing

Chapter 3

Rank and Sign Statistics

3.1 Ranks

3.1.1 Definition

- Suppose we have n observations $\mathbf{X} = (X_1, \ldots, X_n)$. The **rank** R_i of the i^{th} observation is defined as

$$R_i = R_i(\mathbf{X}) = \sum_{j=1}^{n} I(X_i \geq X_j) \tag{3.1}$$

  where

$$I(X_i \geq X_j) = \begin{cases} 1 & \text{if } X_i \geq X_j \\ 0 & \text{if } X_i < X_j \end{cases} \tag{3.2}$$

- The largest observation has a rank of n.
- The smallest observation has a rank of 1 (if there are no ties).
- I am using the notation $R_i(\mathbf{X})$ to emphasize that the rank of the i^{th} observations depends on the entire vector of observations rather than only on the value of X_i.
- You can compute ranks in **R** using the **rank** function:

```r
x <- c(3, 7, 1, 12, 6)  ## 5 observations
rank(x)
```

```
## [1] 2 4 1 5 3
```

3.1.2 Handling Ties

- In the definition of ranks shown in (3.1), tied observations receive their maximum possible rank.
- For example, suppose that $(X_1, X_2, X_3, X_4) = (0, 1, 1, 2)$. In this case, one could argue whether both observations 2 and 3 should be ranked 2^{nd} or 3^{rd} while observations 1 and 4 should unambiguously receive ranks of 1 and 4 respectively.
- Under definition (3.1), both observations 2 and 3 receive a rank of 3.
- In **R**, handling ties in a way that is consistent with definition (3.1) is done using the **ties.method = "max"** argument

```
x <- c(0, 1, 1, 2)
rank(x, ties.method="max")
```

```
## [1] 1 3 3 4
```

- The default in **R** is to replace the ranks of tied observations with their "average" rank

```
x <- c(0, 1, 1, 2)
rank(x)
```

```
## [1] 1.0 2.5 2.5 4.0
```

- As another example of the "average" definition of ranks, consider the following example:

```
y <- c(2, 9, 7, 7, 3, 2, 1)
rank(y, ties.method="max")
```

```
## [1] 3 7 6 6 4 3 1
```

```
rank(y)
```

```
## [1] 2.5 7.0 5.5 5.5 4.0 2.5 1.0
```

- When defining ranks using the "average" or "midrank" approach to handling ties, we replace tied ranks with the average of the two "adjacent" ranks.
- For example, if we have a vector of ranks (R_1, R_2, R_3, R_4) where $R_2 = R_3 = 3$ and $R_1 = 4$ and $R_4 = 1$, then the vector of modified ranks using the "average" approach to handling ties would be

$$(R_1', R_2', R_3', R_4') = \Big(4, \frac{4+1}{2}, \frac{4+1}{2}, 1\Big) \tag{3.3}$$

- The "average" approach is the most common way of handling ties when computing the Wilcoxon rank sum statistic.

3.1.3 Properties of Ranks

Suppose $(X_1, \dots, X_n)$ is random sample from a continuous distribution F (so that the probability of ties is zero). Then, the following properties hold for the associated ranks $R_1, \dots, R_n$.

- Each R_i follows a discrete uniform distribution

$$P(R_i = j) = 1/n, \quad \text{for any } j = 1, \dots, n. \tag{3.4}$$

- The expectation of R_i is

$$E(R_i) = \sum_{j=1}^{n} jP(R_i = j) = \frac{1}{n}\sum_{j=1}^{n} j = \frac{(n+1)}{2} \tag{3.5}$$

- The variance of R_i is

$$\text{Var}(R_i) = E(R_i^2) - E(R_i)^2 = \frac{1}{n}\sum_{j=1}^{n} j^2 - \Big(\frac{n+1}{2}\Big)^2 = \frac{n^2-1}{12} \tag{3.6}$$

- The random variables $R_1, \dots, R_n$ are **not** independent (why?). However, the vector $\mathbf{R}_n = (R_1, \dots, R_n)$ is uniformly distributed on the set of $n!$ permutations of $(1, 2, \dots, n)$.

3.2 The Wilcoxon Rank Sum (WRS) Test: A Two-Sample Test

3.2.1 Goal of the Test

- The Wilcoxon Rank Sum (WRS) test (sometimes referred to as the Wilcoxon-Mann-Whitney test) is a popular, rank-based two-sample test.

- The one-sided WRS test is used to test whether or not observations from one group tend to be larger (or smaller) than observations from the other group.
- Suppose we have observations from two groups: $X_1, \ldots, X_n \sim F_X$ and $Y_1, \ldots, Y_m \sim F_Y$.
- Roughly speaking, the one-sided WRS tests the following hypothesis

$$\begin{aligned} H_0 &: \quad F_X = F_Y \quad \text{versus} \\ H_A &: \quad \text{Observations from } F_X \text{ tend to be larger than observations from } F_Y \end{aligned} \tag{3.7}$$

- What is meant by "tend to be larger" in the alternative hypothesis?
- Two common ways of stating the alternative hypothesis for the WRS include
  1. The stochastic dominance alternative

$$\begin{aligned} H_0 &: \quad F_X = F_Y \quad \text{versus} \\ H_A &: \quad F_X \text{ is stochastically larger than } F_Y \end{aligned} \tag{3.8}$$

  2. The "shift" alternative

$$\begin{aligned} H_0 &: \quad F_X = F_Y \quad \text{versus} \\ H_A &: \quad F_X(t) = F_Y(t - \Delta), \Delta > 0. \end{aligned} \tag{3.9}$$

- A distribution function F_X is said to be stochastically larger than F_Y if $F_X(t) \leq F_Y(t)$ for all t with $F_X(t) < F_Y(t)$ for at least one value of t.
- Note that the "shift alternative" implies stochastic dominance.
- Why do we need to specify an alternative?

- It is often stated that the WRS test is a test of equal medians.
- This is true under the assumption that the relevant alternative is of the form $F_X(t) = F_Y(t - \Delta)$.
- However, one could have a scenario where the two groups have equal medians, but the WRS test has a very high probability of rejecting H_0.
- In addition, in many applications, it is difficult to justify that the "shift alternative" is a reasonable assumption.

- An alternative is to view the WRS test as performing the following hypothesis test:

$$\begin{aligned} H_0: &\quad P(X_i > Y_j) + \tfrac{1}{2}P(X_i = Y_j) = 1/2 \quad \text{versus} \\ H_A: &\quad P(X_i > Y_j) + \tfrac{1}{2}P(X_i = Y_j) > 1/2 \end{aligned} \tag{3.10}$$

  See Divine et al. (2018) for more discussion around this formulation of the WRS test.

- The hypothesis test (3.10) makes fewer assumptions about how F_X and F_Y are related and is, in many cases, more interpretable.

- For example, in medical applications, it is often more natural to answer the question: what is the probability that the outcome under treatment 1 is better than the outcome under treatment 2.

- The justification of hypothesis test (3.10) comes through the close connection between the WRS test statistic W and the Mann-Whitney statistic M. Specifically, $W = M + n(n+1)/2$. (Although, often M is defined as $M = mn + n(n+1)/2 - W$).

- The Mann-Whitney statistic divided by mn is an estimate of the probability:

$$P(X_i > Y_j) + \tfrac{1}{2}P(X_i = Y_j) = 1/2.$$

- The reason for stating H_0 in (3.10) as

$$H_0 : P(X_i > Y_j) + \tfrac{1}{2}P(X_i = Y_j) = 1/2$$

  is to cover the case of either a continuous or discrete distribution.

- When both X_i and Y_j are samples from a continuous distribution we will have $P(X_i = Y_j) = 0$, and we should then think of the null hypothesis as $H_0 : P(X_i > Y_j)$.

- For the case when both X_i and Y_j have a discrete distribution, consider an example where X_i and Y_j have the same discrete distribution with probabilities $P(X_i = 0) = p_0, P(X_i = 1) = p_1$, and $P(X_i = 2) = 1 - p_0 - p_2$.

- With this common discrete distribution on $\{0, 1, 2\}$, we can see that

$P(X_i > Y_j) + \frac{1}{2}P(X_i = Y_j) = 1/2$ because

$$\begin{aligned}
P(X_i > Y_j) + \frac{1}{2}P(X_i = Y_j \quad) &= P(X_i = 1, Y_j = 0) + P(X_i = 2, Y_j = 0) + P(X_i = 2, Y_j = 1) \\
&+ \frac{1}{2}\Big[P(X_i = 0, Y_j = 0) + P(X_i = 1, Y_j = 1) + P(X_i = 2, Y_j = 2)\Big] \\
&= p_1p_0 + (1 - p_1 - p_0)p_0 + (1 - p_1 - p_0)p_1 \\
&+ p_0^2 + p_1^2 + \frac{1}{2} - p_0 - p_1 + p_0p_1 \\
&= 1/2
\end{aligned}$$

3.2.2 Definition of the WRS Test Statistic

- The WRS test statistic is based on computing the sum of ranks (ranks based on the pooled sample) in one group.
- The motivation for the WRS test statistic is the following: if observations from group 1 tend to be larger than those from group 2, the average rank from group 1 should exceed the average rank from group 2.
- A sufficiently large value of the average rank from group 1 will allow us to reject H_0 in favor of H_A.

- We will define the pooled data vector $\mathbf{Z}$ as

$$\mathbf{Z} = (X_1, \dots, X_n, Y_1, \dots, Y_m)$$

  This is a vector with length $n + m$.

- The Wilcoxon rank-sum test statistic W for testing hypotheses of the form (3.8) is then defined as

$$W = \sum_{i=1}^{n} R_i(\mathbf{Z}) \tag{3.11}$$

- In other words, the WRS test statistic is the sum of the ranks for those observations coming from group 1 (i.e., the group with the X_i as observations).
- If the group 1 observations tend to, in fact, be larger than the group 2 observations, then we should expect the sum of the ranks in this group to be larger than the sum of the ranks from group 2.

- Under H_0, we can treat both X_i and Y_i as being observations coming from a common distribution function F.
- Hence, the expectation of $R_i(\mathbf{Z})$ under the null hypothesis is

$$E_{H_0}\{R_i(\mathbf{Z})\} = \frac{n+m+1}{2}$$

and thus the expectation of W under H_0

$$E_{H_0}(W) = \sum_{i=1}^{n} E_{H_0}\{R_i(\mathbf{Z})\} = \frac{n(n+m+1)}{2}$$

- It can be shown that the variance of W under the null hypothesis is

$$\text{Var}_{H_0}(W) = \frac{mn(m+n+1)}{12}$$

3.2.3 Computing p-values for the WRS Test

Exact Distribution

- The p-value for the WRS test is found by computing the probability

$$\text{p-value} = P_{H_0}(W \geq w_{obs}) \tag{3.12}$$

where w_{obs} is the observed WRS test statistic that we get from our data.
- Computing p-values for the WRS test requires us to work with the **null distribution** of W. That is, the distribution of W under the assumption that $F_X = F_Y$.
- The exact null distribution is found by using the fact that each possible ordering of the ranks has the same probability. That is,

$$P\{R_1(\mathbf{Z}) = r_1, \dots, R_{n+m}(\mathbf{Z}) = r_{n+m}\} = \frac{1}{(n+m)!}, \tag{3.13}$$

where $(r_1, \dots, r_{n+m})$ is any permutation of the set $\{1, 2, \dots, n+m\}$. Note that the null distribution only depends on n and m.
- Also, there are $\binom{n+m}{n}$ possible ways to assign distinct ranks to group 1.
- Consider an example with $n = m = 2$. In this case, there are $\binom{4}{2} = 6$ distinct ways to assign 2 ranks to group 1. What is the null distribution of the WRS test statistic? Try to verify that

$$\begin{aligned} P_{H_0}(W = 7) &= 1/6 \\ P_{H_0}(W = 6) &= 1/6 \\ P_{H_0}(W = 5) &= 1/3 \\ P_{H_0}(W = 4) &= 1/6 \\ P_{H_0}(W = 3) &= 1/6. \end{aligned}$$

Large-Sample Approximate Distribution

- Looking at (3.11), we can see that the WRS test statistic is a sum of nearly independent random variables (at least nearly independent for large n and m).
- Thus, we can expect that an appropriately centered and scaled version of W should be approximately Normally distributed (recall the Central Limit Theorem).
- The standardized version $\tilde{W}$ of the WRS is defined as

$$\tilde{W} = \frac{W - E_{H_0}(W)}{\sqrt{\mathrm{Var}_{H_0}(W)}} = \frac{W - n(n+m+1)/2}{\sqrt{mn(n+m+1)/12}} \qquad (3.14)$$

- Under H_0, $\tilde{W}$ converges in distribution to a Normal$(0,1)$ random variable.
- A p-value using this large-sample approximation would then be computed in the following way

$$\begin{aligned} \text{p-value} &= P_{H_0}(W \geq w_{obs}) = P\left(\frac{W - n(n+m+1)/2}{\sqrt{mn(n+m+1)/12}} \geq \frac{w_{obs} - n(n+m+1)/2}{\sqrt{mn(n+m+1)/12}} \right) \\ &= P_{H_0}\Big(\tilde{W} \geq \frac{w_{obs} - n(n+m+1)/2}{\sqrt{mn(n+m+1)/12}} \Big) = 1 - \Phi\left(\frac{w_{obs} - n(n+m+1)/2}{\sqrt{mn(n+m+1)/12}} \right), \end{aligned}$$

  where $\Phi(t)$ denotes the cumulative distribution function of a standard Normal random variable.
- Often, in practice, a continuity correction is applied when using this large-sample approximation. For example, we would compute the probability $P_{H_0}(W \geq w_{obs} - 0.5)$ with the Normal approximation rather than $P_{H_0}(W \geq w_{obs})$ directly.

- Many statistical software packages (including **R**) will not compute p-values using the exact distribution in the presence of ties.
- The **coin** package in **R** does allow you to perform a permutation test in the presence of ties.

- A "two-sided" Wilcoxon rank sum test can also be performed. The two-sided hypothesis tests could either be stated as

$$\begin{aligned} H_0 :& \quad F_X = F_Y \quad \text{versus} \\ H_A :& \quad F_X \text{ is stochastically larger or smaller than } F_Y \end{aligned}$$

or

$$\begin{aligned} H_0 :& \quad F_X = F_Y \quad \text{versus} \\ H_A :& \quad F_X(t) = F_Y(t - \Delta), \Delta \neq 0. \end{aligned}$$

or

$$\begin{aligned} H_0 :& \quad P(X_i > Y_i) + \tfrac{1}{2}P(X_i = Y_i) = 1/2 \quad \text{versus} \\ H_A :& \quad P(X_i > Y_i) + \tfrac{1}{2}P(X_i = Y_i) \neq 1/2 \end{aligned}$$

3.2.4 Computing the WRS test in R

- To illustrate performing the WRS test in **R**, we can use the **wine** dataset from the **rattle.data** package. This dataset is also available from the UCI Machine Learning Repository.

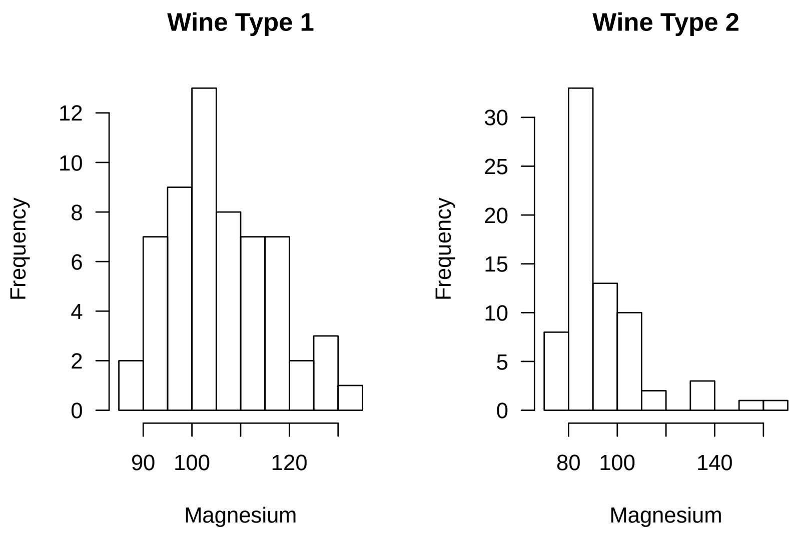
```
library(rattle.data)
head(wine)
```

```
##   Type Alcohol Malic  Ash Alcalinity Magnesium Phenols Flavanoids Nonflavanoids
## 1    1   14.23  1.71 2.43       15.6       127    2.80       3.06          0.28
## 2    1   13.20  1.78 2.14       11.2       100    2.65       2.76          0.26
## 3    1   13.16  2.36 2.67       18.6       101    2.80       3.24          0.30
## 4    1   14.37  1.95 2.50       16.8       113    3.85       3.49          0.24
## 5    1   13.24  2.59 2.87       21.0       118    2.80       2.69          0.39
## 6    1   14.20  1.76 2.45       15.2       112    3.27       3.39          0.34
##   Proanthocyanins Color  Hue Dilution Proline
## 1            2.29  5.64 1.04     3.92    1065
## 2            1.28  4.38 1.05     3.40    1050
## 3            2.81  5.68 1.03     3.17    1185
## 4            2.18  7.80 0.86     3.45    1480
## 5            1.82  4.32 1.04     2.93     735
## 6            1.97  6.75 1.05     2.85    1450
```

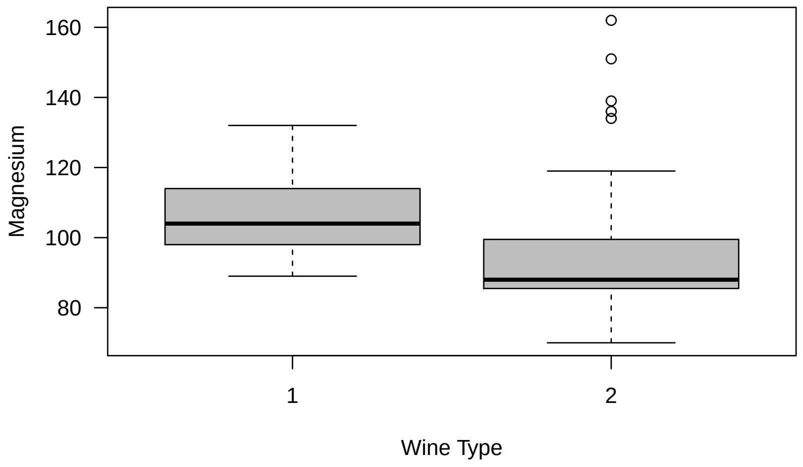
- This dataset contains three types of wine. We will only consider the first two.

```
wine2 <- subset(wine, Type==1 | Type==2)
wine2$Type <- factor(wine2$Type)
```

- Let us consider the difference in the level of magnesium across the two



  types of wine.



- Suppose we are interested in testing whether or not magnesium levels in Type 1 wine are generally larger than magnesium levels in Type 2 wine. This can be done with the following code

```
wilcox.test(x=wine2$Magnesium[wine2$Type==1], y=wine2$Magnesium[wine2$Type==2],
            alternative="greater")
```

```
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  wine2$Magnesium[wine2$Type == 1] and wine2$Magnesium[wine2$Type == 2]
## W = 3381.5, p-value = 8.71e-10
## alternative hypothesis: true location shift is greater than 0
```

- You could also use the following code to perform this test (just be careful about the ordering of the levels of **Type**)

```
wilcox.test(Magnesium ~ Type, data=wine2, alternative="greater")
```

```
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  Magnesium by Type
## W = 3381.5, p-value = 8.71e-10
## alternative hypothesis: true location shift is greater than 0
```

- What is the value of the WRS test statistic? We can code this directly with the following steps:

```
W <- wilcox.test(x=wine2$Magnesium[wine2$Type==1], y=wine2$Magnesium[wine2$Type==2])

n <- sum(wine2$Type==1)
m <- sum(wine2$Type==2)
zz <- rank(wine2$Magnesium) ## vector of pooled ranks
sum(zz[wine2$Type==1])  ## The WRS test statistic
```

```
## [1] 5151.5
```

- The statistic returned by the **wilcox.test** function is actually equal to $W - n(n+1)/2$ not W

```
sum(zz[wine2$Type==1]) - n*(n + 1)/2
```

```
## [1] 3381.5
```

```
W$statistic
```

```
##      W
## 3381.5
```

- $\{W - n(n+1)/2\}$ is equal to the Mann-Whitney statistic. Thus, **W$statistic/(mn)** is an estimate of the probability $P(X_i > Y_j) + P(X_i = Y_j)/2$.

```
W$statistic/(m*n)
```

```
##         W
## 0.8072332
```

- Let's check how the Mann-Whitney statistic matches a simulation-based estimate of this probability

```
ind1 <- which(wine2$Type==1)
ind2 <- which(wine2$Type==2)
xgreater <- rep(0, 100)
for(k in 1:100) {
    xi <- sample(ind1, size=1)
    yi <- sample(ind2, size=1)
    xgreater[k] <- ifelse(wine2$Magnesium[xi] > wine2$Magnesium[yi], 1, 0) +
                   ifelse(wine2$Magnesium[xi] == wine2$Magnesium[yi], 1/2, 0)
}
mean(xgreater)  ## estimate of this probability
```

```
## [1] 0.84
```

- This simulation-based estimate of $P(X_i > Y_j) + P(X_i = Y_j)/2$ is quite close to the value of the Mann-Whitney statistic divided by mn.

3.2.5 Additional Notes for the WRS test

3.2.5.1 Comparing Ordinal Data

- The WRS test is often suggested when comparing categorical data which are **ordinal**.

- For example, we might have 4 categories:

  - Poor
  - Fair
  - Good
  - Excellent

- In this case, there is a natural ordering of the categories but any numerical values assigned to these categories would be arbitrary.
- In such cases, we might be interested in testing whether or not outcomes tend to be better in one group than the other rather than simply comparing whether or not the distribution is different between the two groups.
- A WRS test is useful here since we can still compute ranks without having to choose aribtrary numbers for each category.
- Thinking of the "probability greater than alternative (3.10)" or "stochastically larger than alternative (3.8)" interpretation of the WRS test is probably more reasonable than the "shift alternative (3.9)" interpretation.
- Note that there will probably be many ties when comparing ordinal data.

3.2.5.2 The Hodges-Lehmann Estimator

- The Hodges-Lehmann Estimator $\hat{\Delta}$ is an estimator of Δ in the location-shift model
$$F_X(t) = F_Y(t - \Delta)$$
- The Hodges-Lehmann is defined as the median difference among all possible (group 1, group 2) pairs. Specifically,
$$\hat{\Delta} = \text{median}\{(X_i - Y_j); i = 1, \dots, n; j = 1, \dots, m\}$$
- We won't discuss the Hodges-Lehmann estimator in detail in this course, but in many statistical software packages, the Hodges-Lehmann is often reported when computing the WRS test.
- In **R**, the Hodges-Lehmann estimator can be obtained by using the **conf.int=TRUE** argument in the **wilcox.test** function

```r
WC <- wilcox.test(x=wine2$Magnesium[wine2$Type==1], y=wine2$Magnesium[wine2$Type==2],
                  conf.int=TRUE)
WC$estimate     ## The Hodges-Lehmann estimate
```

```
## difference in location
##                14.00005
```

3.3 One Sample Tests

3.3.1 The Sign Test

3.3.1.1 Motivation and Definition

- The **sign test** can be thought of as a test of whether or not the median of a distribution is greater than zero (or greater than some other fixed value θ_0).
- Frequently, the sign test is applied in the following context:
  - Suppose we have observations $D_1, \dots, D_n$ which arise from the model
  $$D_i = \theta + \varepsilon_i, \tag{3.15}$$
  where ε_i are iid random variables each with distribution function F_ϵ that is assumed to have a median of zero. Moreover, we will assume the density function $f_\varepsilon(t)$ of ε_i is symmetric around zero.
- The distribution function of D_i is then
$$F_D(t) = P(D_i \leq t) = P(\varepsilon_i \leq t - \theta) = F_\epsilon(t - \theta) \tag{3.16}$$
- Likewise, the density function $f_D(t)$ of D_i is given by
$$f_D(t) = f_\epsilon(t - \theta) \tag{3.17}$$
- In this context, θ is usually referred to as a **location parameter**.
- The goal here is to test $H_0 : \theta = \theta_0$ vs. $H_A : \theta > \theta_0$. (Often, $\theta_0 = 0$).

- This sort of test usually comes up in the context of **paired data**. Common examples include
  - patients compared "pre and post treatment"
  - students before and after the introduction of a new teaching method
  - comparison of "matched" individuals who are similar (e.g., same age, sex, education, etc.)
  - comparing consistency of measurements made on the same objects
- In such cases, we have observations X_i and Y_i for $i = 1, \dots n$ where it is not necessarily reasonable to think of X_i and Y_i as independent.

	Baseline_Measure	Post_Treatment_Measure
Patient 1	Y1	X1
Patient 2	Y2	X2
Patient 3	Y3	X3
Patient 4	Y4	X4

- We can define $D_i = X_i - Y_i$ as the difference in the i^{th} pair.
- With this setup, a natural question is whether or not the differences D_i tend to be greater than zero or not.

- The **sign** statistic S_n is defined as

$$S_n = \sum_{i=1}^{n} I(D_i > 0) \tag{3.18}$$

- If the null hypothesis $H_0 : \theta = 0$ is true, then we should expect that roughly half of the observations will be positive.
- This suggests that we will reject H_0 if $S_n \geq c$, where c is a number that is greater than $n/2$.

3.3.1.2 Null Distribution and p-values for the Sign Test

- Notice that the sign statistic defined in (3.18) is the sum of independent Bernoulli random variable.
- That is, we can think of $Z_i = I(D_i > 0)$ as a random variable with success probability $p(\theta)$ where the formula for $p(\theta)$ is

$$p(\theta) = P(Z_i = 1) = P(D_i > 0) = 1 - F_D(0) = 1 - F_\epsilon(-\theta)$$

- This implies that S_n is a binomial random variable with n trials and success probability $p(\theta)$. That is,

$$S_n \sim \text{Binomial}(n, p(\theta)) \tag{3.19}$$

- Because $p(0) = 1/2$, $S_n \sim \text{Binomial}(n, 1/2)$ under H_0.
- Notice that the "null distribution" of the sign statistic is "distribution free" in the sense that the null distribution of S_n does not depend on the distribution of D_i.

- The p-value for the one-sided sign test can be computed by

$$\text{p-value} = P_{H_0}(S_n \geq s_{obs}) = \sum_{j=s_{obs}}^{n} P_{H_0}(S_n = j) = \sum_{j=s_{obs}}^{n} \binom{n}{j}\frac{1}{2^n},$$

  where s_{obs} is the observed value of the sign statistic.

```r
### How to compute the p-value for the sign test using R
xx <- rnorm(100)
sign.stat <- sum(xx > 0)  ## This is the value of the sign statistic
1 - pbinom(sign.stat - 1, size=100, prob=1/2) ## p-value for sign test
```

```
## [1] 0.4602054
```

- The reason that this is the right expression using **R** is that for any positive integer w

$$P_{H_0}(S_n \geq w) = 1 - P_{H_0}(S_n < w) = 1 - P_{H_0}(S_n \leq w - 1) \tag{3.20}$$

  and the **R** function **pbinom(t, n, prob)** computes $P(X \leq t)$ where X is a binomial random variable with n trials and success probability **prob**.

- You can also perform the one-sided sign test by using the **binom.test** function in **R**.

```r
btest <- binom.test(sign.stat, n=100, p=0.5, alternative="greater")
btest$p.value
```

```
## [1] 0.4602054
```

3.3.1.3 Two-sided Sign Test

- Notice that the number of negative values of D_i can be expressed as

$$\sum_{i=1}^{n} I(D_i < 0) = n - S_n \tag{3.21}$$

  if there are no observations that equal zero exactly. Large value of $n - S_n$ would be used in favor of another possible one-sided alternative $H_A : \theta < 0$.

- If we now want to test the two-sided alternative

$$H_0 : \theta = 0 \quad \text{vs.} \quad H_A : \theta \neq 0$$

  you would need to compute the probability under the null hypothesis of observing a "more extreme" observation than the one that was actually observed.

- Extreme is defined by thinking about the fact that we would have rejected H_0 if either S_n or $n - S_n$ were very large.
- For example, if $n = 12$, then the expected value of the sign statistic would be 6. If $s_{obs} = 10$, then the collection of "more extreme" events would be ≤ 2 or ≥ 10.
- The two-sided p-value is determined by looking at the tail probabilities on both sides

$$\text{p-value} = \begin{cases} P_{H_0}(S_n \geq s_{obs}) + P_{H_0}(S_n \leq n - s_{obs}) & \text{if } s_{obs} \geq n/2 \\ P_{H_0}(S_n \leq s_{obs}) + P_{H_0}(S_n \geq n - s_{obs}) & \text{if } s_{obs} < n/2 \end{cases}$$

- It actually works out that

$$\text{p-value} = \begin{cases} 2P_{H_0}(S_n \geq s_{obs}) & \text{if } s_{obs} \geq n/2 \\ 2P_{H_0}(S_n \leq s_{obs}) & \text{if } s_{obs} < n/2 \end{cases}$$

- Also, you can note that this p-value would be the same that you would get from performing the test $H_0 : p = 1/2$ vs. $H_A : p \neq 1/2$ when it is assumed that $S_n \sim \text{Binomial}(n, p)$.
- Another note: It is often suggested that one should drop observations which are exactly zero when performing the sign test.

3.3.2 The Wilcoxon Signed Rank Test

- The Wilcoxon signed rank test can be applied under the same scenario that we used the sign test.
- One criticism of the sign test is that it ignores the magnitude of the observations.
- For example, the sign test statistic S treats observations $D_i = 0.2$ and $D_i = 3$ the same.
- The **Wilcoxon signed rank statistic** T_n weights the signs of D_i by the rank of its absolute value.
- Specifically, the Wilcoxon signed rank statistic is defined as

$$T_n = \sum_{i=1}^{n} \text{sign}(D_i) R_i(|\mathbf{D}|)$$

where the sign function is defined as

$$\text{sign}(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{if } x = 0 \\ -1 & \text{if } x < 0 \end{cases}$$

- Here, $R_i(|\mathbf{D}|)$ is the rank of the i^{th} element from the vector $|\mathbf{D}| = (|D_1|, |D_2|, \dots, |D_n|)$.
- Intuitively, the Wilcoxon signed rank statistic is measuring whether or not large values of $|D_i|$ tend to be associated with positive vs. negative values of D_i.

3.3.2.1 Asymptotic Distribution

- As mentioned in the above exercise, the expectation of T_n under H_0 is zero.
- It can be shown that the variance under the null hypothesis is

$$\text{Var}_{H_0}(T_n) = \frac{n(2n+1)(n+1)}{6}$$

- Similar, to the large-sample approximation we used for the WRS test, we have the following asymptotic result for the Wilcoxon signed-rank test

$$\frac{T_n}{\sqrt{\text{Var}_{H_0}(T_n)}} \longrightarrow \text{Normal}(0,1) \quad \text{as } n \longrightarrow \infty$$

- Because the variance of T is dominated by the term $n^3/3$ for very large n, we could also say that under H_0 that

$$\frac{T_n}{\sqrt{n^3/3}} \longrightarrow \text{Normal}(0,1) \quad \text{as } n \longrightarrow \infty \tag{3.22}$$

  In other words, we can say that T_n has an approximately Normal$(0, n^3/3)$ for large n.

3.3.2.2 Exact Distribution

- The exact distribution of the Wilcoxon signed rank statistic T_n is somewhat more complicated than the exact distribution of the WRS test statistic. Nevertheless, there exists functions in **R** for working with this exact distribution.

3.3.3 Using R to Perform the Sign and Wilcoxon Tests

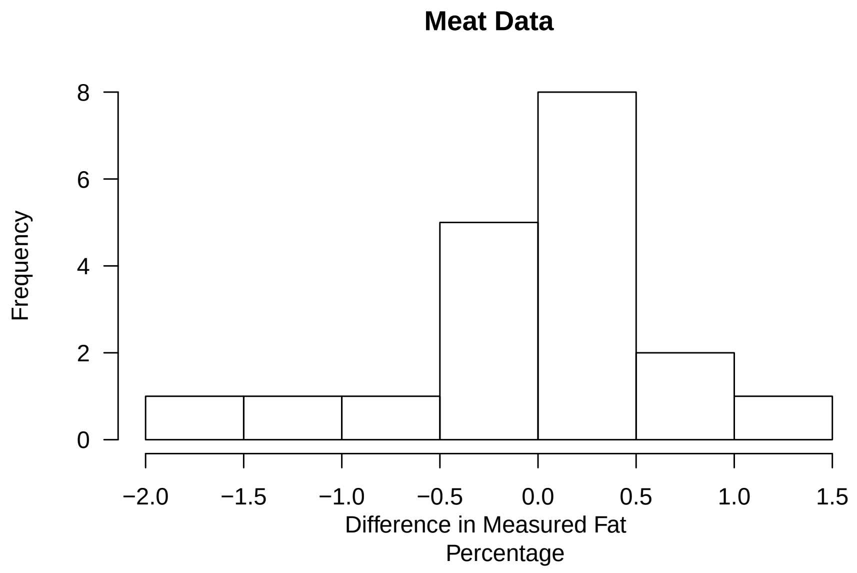
- Let's first look at the **Meat** data from the **PairedData R** package.
- This data set contains 20 observations with each observation corresponding to a single piece of meat. For each observation, we have two measures of fat percentage that were obtained different measuring techniques.

```
library(PairedData, quietly=TRUE, warn.conflicts=FALSE) ## loading PairedData package
data(Meat)  ## loading Meat data
head(Meat)
```

```
##   AOAC Babcock     MeatType
## 1 22.0    22.3       Wiener
## 2 22.1    21.8       Wiener
## 3 22.1    22.4       Wiener
## 4 22.2    22.5       Wiener
## 5 24.6    24.9   ChoppedHam
## 6 25.3    25.6 ChooppedPork
```

- Define the differences D_i as the **Babcock** measurements minus the **AOAC** measures. We will drop the single observation that equals zero.

```
DD <- Meat[,2] - Meat[,1]
DD <- DD[DD!=0]
hist(DD, main="Meat Data", xlab="Difference in Measured Fat
     Percentage", las=1)
```



```
summary(DD)
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -1.60000 -0.25000  0.30000  0.04211  0.40000  1.10000
```

The Sign Test in R

- Let's first test the hypothesis $H_0 : \theta = 0$ vs. $H_A : \theta \neq 0$ using the two-sided sign test. This can be done using the **binom.test** function

```
binom.test(sum(DD > 0), n = length(DD), p=0.5)$p.value
```

```
## [1] 0.6476059
```

Wilcoxon Signed Rank Test in R

- You can actually use the function **wilcox.test** to perform the Wilcoxon signed rank test in addition to the Wilcoxon rank sum test. To perform the Wilcoxon signed rank test in **R**, you just need to enter data for the **x** argument and leave the **y** argument empty.

```
wilcox.test(x=DD)
```

```
## Warning in wilcox.test.default(x = DD): cannot compute exact p-value with ties
```

```
##
##  Wilcoxon signed rank test with continuity correction
##
## data:  DD
## V = 118.5, p-value = 0.3534
## alternative hypothesis: true location is not equal to 0
```

- You will note that the p-value for the Wilcoxon signed rank test is lower than that of the sign test. In general, the Wilcoxon signed rank test is somewhat more "sensitive" than the sign test meaning that it will have a greater tendency to reject H_0 for small deviations from H_0.
- We can explore this sensitivity comparison with a small simulation study. We will consider a scenario where $D_i = 0.4 + \varepsilon_i$ with ε_i having a t distribution with 3 degrees of freedom.

```
set.seed(1327)
n.reps <- 500  ## number of simulation replications
samp.size <- 50  ## the sample size
wilcox.reject <- rep(0, n.reps)
sign.reject <- rep(0, n.reps)
for(k in 1:n.reps) {
    dsim <- .4 + rt(samp.size, df=3)
    wilcox.reject[k] <- ifelse(wilcox.test(x=dsim)$p.value < 0.05, 1, 0)
    sign.reject[k] <- ifelse(binom.test(sum(dsim > 0),
                                     n=samp.size, p=0.5)$p.value < 0.05, 1, 0)
}
mean(wilcox.reject)  ## proportion of times Wilcoxon signed rank rejected H0
```

```
## [1] 0.614
```

```
mean(sign.reject)  ## proportion of times Wilcoxon signed rank rejected H0
```

```
## [1] 0.488
```

3.4 Power and Comparisons with Parametric Tests

3.4.1 The Power Function of a Test

- The **power** of a test is the probability that a test rejects the null hypothesis when the alternative hypothesis is true.
- The alternative hypothesis H_A is usually characterized by a large range of values of the parameter of interest. For example, $H_A : \theta > 0$ or $H_A : \theta \neq 0$.
- For this reason, it is better to think of power as a function that varies across the range of the alternative hypothesis.
- To be more precise, we will define the power function as a function of some parameter θ where the null hypothesis corresponds to $\theta = \theta_0$ and the alternative hypothesis represents a range of alternative values of θ.
- The power function $\gamma_n(\cdot)$ of a testing procedure is defined as

$$\gamma_n(\delta) = P_{\theta=\delta}\{\text{reject } H_0\} \qquad \text{for } \delta \in H_A.$$

- The notation $P_{\theta=\delta}\{\text{reject } H_0\}$ means that we are computing this probability under the assumption that the parameter of interest θ equals δ.

The Approximate Power Function of the Sign Test

- Let us consider the sign test for testing $H_0 : \theta = 0$ vs. $\theta > 0$.

- The sign test is based on the value of the sign statistic S_n.

- Recalling (3.19), we know that $S_n \sim \text{Binomial}(n, p(\theta))$. Hence,

$$\sqrt{n}\left(\tfrac{S_n}{n} - p(\theta)\right) \longrightarrow \text{Normal}\Big(0, p(\theta)(1-p(\theta))\Big) \quad \text{as } n \longrightarrow \infty \tag{3.23}$$

- The sign test will reject H_0 when $S_n \geq c_{\alpha,n}$ where the constant $c_{\alpha,n}$ is chosen so that $P_{H_0}(S_n \geq c_{\alpha,n}) = \alpha$. Using the large-sample approximation (3.23), you can show that

$$c_{\alpha,n} = \frac{n + \sqrt{n} z_{1-\alpha}}{2}, \tag{3.24}$$

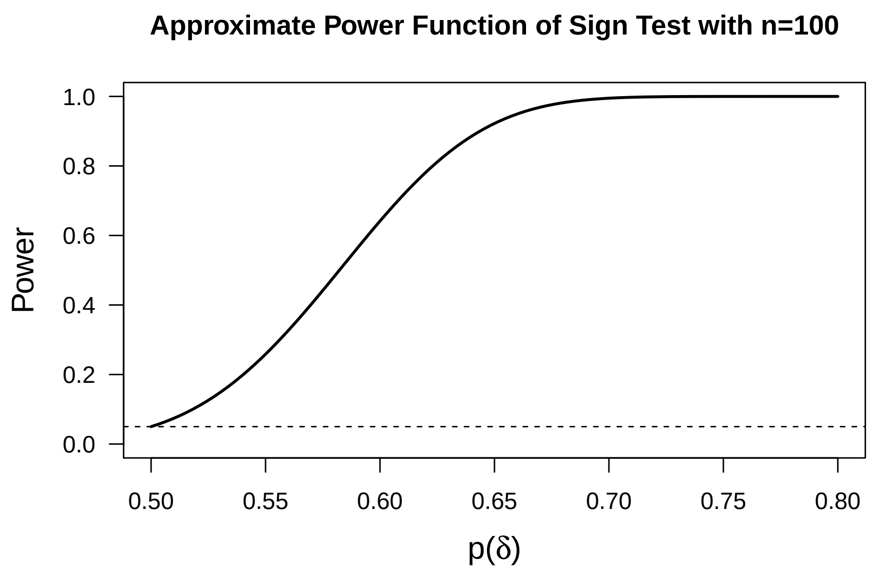
  where $z_{1-\alpha}$ denotes the upper $1-\alpha$ quantile of the standard normal distribution. In other words, $\Phi(z_{1-\alpha}) = 1-\alpha$.

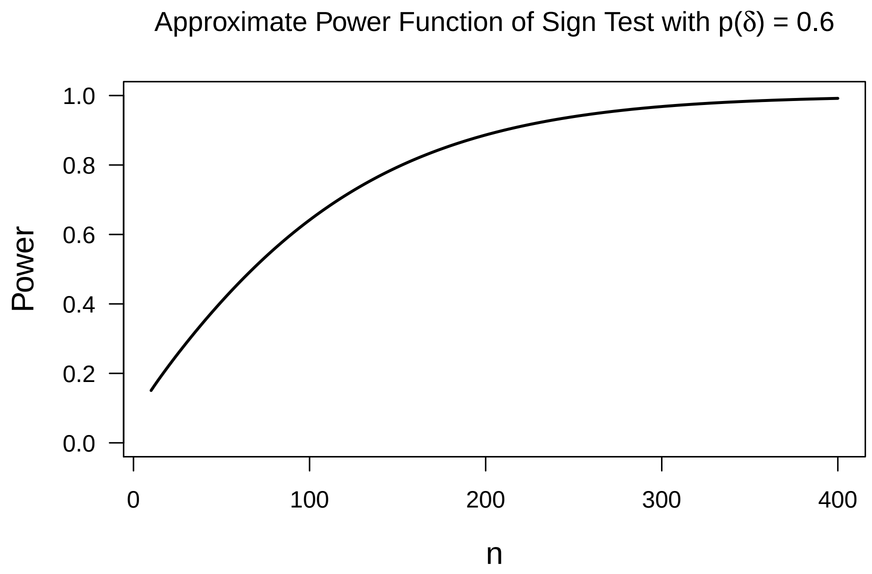
- Also, when using large-sample approximation (3.23), the power of this test to detect a value of $\theta = \delta$ is given by

$$\begin{aligned}
\gamma_n(\delta) &= P_{\theta=\delta}\{S_n \geq c_{\alpha,n}\} = P_{\theta=\delta}\left\{\frac{\sqrt{n}(S_n/n - p(\delta))}{\sqrt{p(\delta)(1-p(\delta))}} \geq \frac{\sqrt{n}(c_{\alpha,n}/n - p(\delta))}{\sqrt{p(\delta)(1-p(\delta))}}\right\} \\
&= 1 - \Phi\left(\frac{\sqrt{n}(c_{\alpha,n}/n - p(\delta))}{\sqrt{p(\delta)(1-p(\delta))}}\right) \\
&= 1 - \Phi\left(\frac{z_{1-\alpha}}{2\sqrt{p(\delta)(1-p(\delta))}} - \frac{\sqrt{n}(p(\delta)-1/2)}{\sqrt{p(\delta)(1-p(\delta))}}\right) \qquad (3.25)
\end{aligned}$$

- Notice that the power of the test depends more directly on the term $p(\delta) = P_{\theta=\delta}(D_i > 0)$. Recall from Section 3.3.1 that $p(\delta) = 1 - F_\epsilon(-\delta)$, where F_ϵ is the distribution function of ε_i in the model $D_i = \theta + \varepsilon_i$.

- So, in any power or sample size calculation, it would be more sensible to think about plausible values for $p(\delta)$ rather than δ itself. Plus, $p(\delta)$ has the direct interpretation $p(\delta) = P_{\theta=\delta}(D_i > 0)$.


Approximate Power Function of Sign Test with n=100
Power
p(δ)
1.0
0.8
0.6
0.4
0.2
0.0
0.50
0.55
0.60
0.65
0.70
0.75
0.80


Approximate Power Function of Sign Test with p(δ) = 0.6
Power
n
1.0
0.8
0.6
0.4
0.2
0.0
0
100
200
300
400

3.4.2 Power Comparisons and Asymptotic Relative Efficiency

- Notice that for the sign statistic power function shown in (3.25), we have that
$$\lim_{n \longrightarrow \infty} \gamma_n(\delta) = \begin{cases} \alpha & \text{if } \delta = 0 \\ 1 & \text{if } \delta > 0 \end{cases} \tag{3.26}$$
- The above type of limit for the power function is will be true for most "reasonable" tests.
- Indeed, a test whose power function satisfies (3.26) is typically called a **consistent** tests.

- If nearly all reasonable tests are consistent, then how can we compare tests with respect to their power?
- One approach is to use simulations to compare power for several plausible alternatives. While this can be useful for a specific application, it limits our ability to make more general statements about power comparisons.
- Another approach might be to determine for which values of (δ, n) one test has greater power than another. However, this could be tough to interpret (no test will be uniformly more powerful for all distributions) or even difficult to compute.
- One way to think about power is to think about the **relative efficiency** of two testing procedures. The efficiency of a test in this context is the sample size required to achieve a certain level of power.

- To find the asymptotic relative efficiency, we first need to derive the asymptotic power function.
- For our hypothesis $H_0 : \theta = \theta_0$ vs. $H_A : \theta > \theta_0$, this is defined as
$$\tilde{\gamma}(\delta) = \lim_{n \longrightarrow \infty} \gamma_n(\theta_0 + \delta/\sqrt{n})$$
- Considering the sequence of "local alternatives" $\theta_n = \theta_0 + \delta/\sqrt{n}$, we avoid the problem of the power always converging to 1.

- It can be shown that

$$\tilde{\gamma}(\delta) = 1 - \Phi\left(z_{1-\alpha} - \delta \frac{\mu'(\theta_0)}{\sigma(\theta_0)} \right) \tag{3.27}$$

as long as we can find functions $\mu(\cdot)$ and $\sigma(\cdot)$ such that

$$\frac{\sqrt{n}(V_n - \mu(\theta_n))}{\sigma(\theta_n)} \longrightarrow \text{Normal}(0,1) \tag{3.28}$$

where the test of $H_0 : \theta = \theta_0$ vs. $H_A : \theta > \theta_0$ is based on the test statistic V_n with rejection of H_0 occurring whenever $V_n \geq c_{\alpha,n}$. Statement (3.28) asssumes that the distribution of V_n is governed by θ_n for each n.

- The ratio $e(\theta_0) = \mu'(\theta_0)/\sigma(\theta_0)$ is the **asymptotic efficiency** of the test.
- When comparing two tests with efficiency $e_1(\theta_0)$ and $e_2(\theta_0)$, the asymptotic relative efficiency of test 1 vs. test 2 is defined as

$$ARE_{12}(\theta_0) = \Big(\frac{e_1(\theta_0)}{e_2(\theta_0)} \Big)^2 \tag{3.29}$$

Interpretation of Asymptotic Efficiency of Tests

- Roughly speaking, the asymptotic relative efficiency $ARE_{12}(\theta_0)$ approximately equals n_2/n_1 where n_1 is the sample size needed for test 1 to achieve power β and n_2 is the sample size needed for test 2 to achieve power β. This is true for an arbitrary β.
- To further justify this interpretation notice that, for large n, we should have

$$c_{\alpha,n} \approx \mu(\theta_0) + \frac{\sigma(\theta_0) z_{1-\alpha}}{\sqrt{n}} \tag{3.30}$$

(This approximation for $c_{\alpha,n}$ comes from the asymptotic statement in (3.28))
- Now, consider the power for detecting $H_A : \theta = \theta_A$ (where we will assume that θ_A is "close" to θ_0). Using (3.28), the approximate power in this setting is

$$\begin{aligned} P_{\theta_A}\Big(V_n \geq c_{\alpha,n} \Big) &= P_{\theta_A}\left(\frac{\sqrt{n}(V_n - \mu(\theta_A))}{\sigma(\theta_A)} \geq \frac{\sqrt{n}(c_{\alpha,n} - \mu(\theta_A))}{\sigma(\theta_A)} \right) \\ &\approx 1 - \Phi\left(\frac{\sqrt{n}(c_{\alpha,n} - \mu(\theta_A))}{\sigma(\theta_A)} \right) \\ &= 1 - \Phi\left(\frac{\sqrt{n}(\mu(\theta_0) - \mu(\theta_A))}{\sigma(\theta_A)} + \frac{z_{1-\alpha}\sigma(\theta_0)}{\sigma(\theta_A)} \right) \end{aligned} \tag{3.31}$$

- Hence, if we want to achieve a power level of β for the alternative $H_A : \theta = \theta_A$, we need the corresponding sample size $n_\beta(\theta_A)$ to satisfy

$$\frac{\sqrt{n_\beta(\theta_A)}(\mu(\theta_0) - \mu(\theta_A))}{\sigma(\theta_A)} + \frac{z_{1-\alpha}\sigma(\theta_0)}{\sigma(\theta_A)} = z_{1-\beta} \tag{3.32}$$

which reduces to

$$n_\beta(\theta_A) = \left(\frac{z_{1-\beta}\sigma(\theta_A) - z_{1-\alpha}\sigma(\theta_0)}{\mu(\theta_0) - \mu(\theta_A)}\right)^2 \approx \left(\frac{[z_{1-\beta} - z_{1-\alpha}]\sigma(\theta_0)}{(\theta_A - \theta_0)\mu'(\theta_0)}\right)^2 \tag{3.33}$$

- So, if we were comparing two testing procedures and we computed the approximate sample sizes $n_\beta^1(\theta_A)$ and $n_\beta^2(\theta_A)$ needed to reach β power for the alternative $H_A : \theta = \theta_A$, the sample size ratio (using approximation (3.33)) would be

$$\frac{n_\beta^2(\theta_A)}{n_\beta^1(\theta_A)} = \left(\frac{\mu_1'(\theta_0)\sigma_2(\theta_0)}{\mu_2'(\theta_0)\sigma_1(\theta_0)}\right)^2 = \text{ARE}_{12}(\theta_0) \tag{3.34}$$

- Notice that $\text{ARE}_{12}(\theta_0) > 1$ indicates that the test 1 is better than test 2 because the sample size required for test 1 would be less than the sample size required for test 2.

- It is also worth noting that our justification for the interpretation of $\text{ARE}_{12}(\theta_0)$ was not very rigorous or precise, but it is possible to make a more rigorous statement. See, for example, Chapter 13 of Lehmann and Romano (2006) for a more rigorous treatment of relative efficiency.

- In Lehmann and Romano (2006), they have a result that states (under appropriate assumptions) that

$$\lim_{\theta \downarrow \theta_0} \frac{N_2(\theta)}{N_1(\theta)} = ARE_{12}(\theta_0) \tag{3.35}$$

where $N_1(\theta)$ and $N_2(\theta)$ are the sample sizes required to have power β against alternative θ.

3.4.3 Efficiency Examples

The Sign Test

- Let us return to the example of the sign statistic S_n and its use in testing the hypothesis $H_0 : \theta = 0$ vs. $H_A : \theta > 0$.

- Notice that the sign test rejects $H_0 : \theta = 0$ for $V_n > c_{\alpha,n}$ where $V_n = S_n/n$ and S_n is the sign statistic.
- When V_n is defined this way (3.28) is satisfied when $\mu(\theta) = p(\theta)$ and $\sigma(\theta) = \sqrt{p(\theta)(1-p(\theta))}$ where $p(\theta) = 1 - F_\epsilon(-\theta)$.
- Thus, the efficiency of the sign test for testing $H_0 : \theta = 0$ vs. $H_A : \theta > 0$ is
$$\frac{\mu'(0)}{\sigma(0)} = \frac{p'(0)}{\sqrt{p(0)(1-p(0))}} = 2f_\epsilon(0)$$
where $f_\epsilon(t) = F'_\epsilon(t)$.

The One-Sample t-test

- Assume that we have data $D_1, \dots, D_n$ generated under the same assumption as in our discussion of the sign test and the Wilcoxon signed-rank test. That is,
$$D_i = \theta + \varepsilon_i,$$
where ε_i are assumed to have median 0 with ε_i having p.d.f. f_ε
- The one-sample t-test will reject $H_0 : \theta = 0$ whenever $V_n > c_{\alpha,n}$, where V_n is defined to be
$$V_n = \frac{\bar{D}}{\hat{\sigma}}$$
- Note that (3.28) will apply if we choose
$$\begin{aligned} \mu(\theta) &= E_\theta(D_i) = \theta \\ \sigma(\theta) &= \sqrt{\mathrm{Var}_\theta(D_i)} = \sqrt{\mathrm{Var}(\varepsilon_i)} = \sigma_\epsilon \end{aligned}$$
- These choices of $\mu(\theta)$ and $\sigma(\theta)$ work because
$$\begin{aligned} \frac{\sqrt{n}(V_n - \mu(\theta_n))}{\sigma(\theta_n)} &= \frac{\sqrt{n}(\bar{D} - \theta_n)}{\sigma_e} + \sqrt{n}\theta_n\Big(\frac{1}{\hat{\sigma}} - \frac{1}{\sigma_e}\Big) \\ &= \frac{\sqrt{n}(\bar{D} - \theta_n)}{\sigma_e} + \delta\Big(\frac{1}{\hat{\sigma}} - \frac{1}{\sigma_e}\Big) \\ &\longrightarrow \mathrm{Normal}(0,1) \end{aligned}$$
- So, the efficiency of the one-sample t-test is given by
$$\frac{\mu'(0)}{\sigma(0)} = \frac{1}{\sigma_e}$$

The Wilcoxon Rank Sum Test

- Using the close relation between the WRS test statistic and the Mann-Whitney statistic, the WRS test can be represented as rejecting H_0 when $V_N \geq c_{\alpha,N}$ where V_N is

$$V_N = \frac{1}{mn}\sum_{i=1}^{n}\sum_{j=1}^{m} I(X_i \geq Y_j)$$

  and $N = n + m$.

- The power of the WRS test is usually analyzed in the context of the "shift alternative". Namely, we are assuming that $F_X(t) = F_Y(t - \theta)$ and test $H_0 : \theta = 0$ vs. $H_A : \theta > 0$.
- The natural choice for $\mu(\theta)$ is the expectation of V_N when θ is the true shift parameter.
- So, let $\mu(\theta) = P_\theta(X_i \geq Y_j)$. This can be written in terms of F_Y and f_Y:

$$\begin{aligned}\mu(\theta) &= \int_{-\infty}^{\infty} P_\theta(X_i \geq Y_j|Y_j = t)f_Y(t)dt = \int_{-\infty}^{\infty} P_\theta(X_i \geq t)f_Y(t)dt \\ &= \int_{-\infty}^{\infty}\{1 - F_X(t)\}f_Y(t)dt = 1 - \int_{-\infty}^{\infty} F_Y(t-\theta)f_Y(t)dt \quad (3.36)\end{aligned}$$

- You can show that (3.28) holds (see e.g, Chapter 14 of Van der Vaart (2000)) if you choose $\sigma^2(\theta)$ to be

$$\sigma^2(\theta) = \frac{1}{1-\lambda}\text{Var}\{F_Y(X_i)\} + \frac{1}{\lambda}\text{Var}\{F_Y(Y_i - \theta)\} \qquad (3.37)$$

  Here, $n/(m+n) \longrightarrow \lambda$.

- Thus, the efficiency of testing $H_0 : \theta = 0$ for the WRS test is

$$e(0) = \frac{\mu'(0)}{\sigma(0)} = \frac{\int_{-\infty}^{\infty} f^2(t)dt}{\sigma(0)} \qquad (3.38)$$

3.4.4 Efficiency Comparisons for Several Distributions

Sign Test vs. One-Sample t-test

- Comparisons of the Efficiency of the sign and one-sample t-test only require us to find $f_\epsilon(0)$ and σ_e^2 for different assumptions about the residual density f_ϵ.

- For the Logistic(0,1) distribution, $f_\epsilon(0) = 1/4$ and the standard deviation is $\pi/\sqrt{3}$. Hence, the asymptotic relative efficiency of the sign test vs. the one-sample t-test is $(\pi/2\sqrt{3})^2 = \pi^2/12$.
- The relative efficiencies for the sign vs. t-test for other distributions are shown below

Distribution	Efficiency	(3.39)
$\text{Normal}(0,1)$	$2/\pi$	(3.40)
$\text{Logistic}(0,1)$	$\pi^2/12$	(3.41)
$\text{Laplace}(0,1)$	2	(3.42)
$\text{Uniform}(-1,1)$	$1/3$	(3.43)
t-dist_ν	$[4(\nu/(\nu-2))\Gamma^2\{(\nu+1)/2\}]/[\Gamma^2(\nu/2)\nu\pi]$	(3.44)

WRS Test vs. Two-Sample t-test

- The relative efficiencies for the WRS test vs. the two-sample t-test for several distributions are shown below.

Distribution	Efficiency	(3.45)
$\text{Normal}(0,1)$	$3/\pi$	(3.46)
$\text{Logistic}(0,1)$	$\pi^2/9$	(3.47)
$\text{Laplace}(0,1)$	$3/2$	(3.48)
$\text{Uniform}(-1,1)$	1	(3.49)
t-dist_3	1.24	(3.50)
t-dist_5	1.90	(3.51)
		(3.52)

3.4.5 A Power "Contest"

- To compare power for specific sample sizes, effect sizes, and distributional assumptions, a simulation study can be more helpful than statements about asymptotic relative efficiency.
- Below shows the results of a simulation study in **R** which compares power for the one-sample testing problem.
- This simulation study compares the sign test, Wilcoxon signed rank test, and the one-sample t-test.

- It is assumed that $n = 200$ and that responses D_i are generated from the following model:

$$D_i = 0.2 + \varepsilon_i$$

- Three choices for the distribution of ε_i were considered:
  - $\varepsilon_i \sim \text{Logistic}(0, 1)$
  - $\varepsilon_i \sim \text{Normal}(0, 1)$
  - $\varepsilon_i \sim \text{Uniform}(-3/2, 3/2)$

- The **R** code and simulation results are shown below.

```r
set.seed(148930)
theta <- 0.2
n <- 200
nreps <- 500
RejectSign <- RejectWilcoxonSign <- RejectT <- matrix(NA, nrow=nreps, ncol=4)
for(k in 1:nreps) {
  xx <- theta + rlogis(n)
  yy <- theta + rnorm(n)
  zz <- theta + runif(n, min=-3/2, max=3/2)
  ww <- theta + (rexp(n, rate=1) - rexp(n, rate=1))/sqrt(2)

  RejectSign[k,1] <- ifelse(binom.test(x=sum(xx > 0), n=n, p=0.5)$p.value < 0.05, 1, 0
  RejectWilcoxonSign[k,1] <- ifelse(wilcox.test(xx)$p.value < 0.05, 1, 0)
  RejectT[k,1] <- ifelse(t.test(xx)$p.value < 0.05, 1, 0)

  RejectSign[k,2] <- ifelse(binom.test(x=sum(yy > 0), n=n, p=0.5)$p.value < 0.05, 1, 0
  RejectWilcoxonSign[k,2] <- ifelse(wilcox.test(yy)$p.value < 0.05, 1, 0)
  RejectT[k,2] <- ifelse(t.test(yy)$p.value < 0.05, 1,0)

  RejectSign[k,3] <- ifelse(binom.test(x=sum(zz > 0), n=n, p=0.5)$p.value < 0.05, 1, 0
  RejectWilcoxonSign[k,3] <- ifelse(wilcox.test(zz)$p.value < 0.05, 1, 0)
  RejectT[k,3] <- ifelse(t.test(zz)$p.value < 0.05,1,0)

  RejectSign[k,4] <- ifelse(binom.test(x=sum(ww > 0), n=n, p=0.5)$p.value < 0.05, 1, 0
  RejectWilcoxonSign[k,4] <- ifelse(wilcox.test(ww)$p.value < 0.05, 1, 0)
  RejectT[k,4] <- ifelse(t.test(ww)$p.value < 0.05, 1, 0)
}

power.results <- data.frame(Distribution=c("Logistic", "Normal", "Uniform", "Laplace")
                  SignTest=colMeans(RejectSign), WilcoxonSign=colMeans(RejectWilcoxonSi
                  tTest=colMeans(RejectT))
```

Distribution	SignTest	WilcoxonSign	tTest
Logistic	0.25	0.37	0.34
Normal	0.59	0.77	0.81
Uniform	0.44	0.87	0.90
Laplace	0.93	0.92	0.81

Table 3.1: Estimated power for three one-sample tests and three distributions. 500 simulation replications were used.

3.5 Linear Rank Statistics in General

3.5.1 Definition

- The Wilcoxon rank sum statistic is an example of a statistic that belongs to a more general class of rank statistics.
- This is the class of **linear rank statistics**.
- Suppose we have observations $\mathbf{Z} = (Z_1, \dots, Z_N)$. A linear rank statistic is a statistic T_N that can be expressed as

$$T_N = \sum_{i=1}^{N} c_{iN} a_N(R_i(\mathbf{Z})) \tag{3.53}$$

- The terms $c_{1N}, \dots, c_{NN}$ are usually called **coefficients**. These are fixed numbers and are not random variables.
- The terms $a_N(R_i(\mathbf{Z}))$ are commonly referred to as **scores**.
- Typically, the scores are generated from a given function ψ in the following way

$$a_N(i) = \psi\Big(\frac{i}{N+1}\Big)$$

Example: WRS statistic

- For the Wilcoxon rank sum test, we separated the data $\mathbf{Z} = (Z_1, \dots, Z_N)$ into two groups.
- The first n observations were from group 1 while the last m observations were from group 2.
- The WRS statistic was then defined as

$$W = \sum_{i=1}^{n} R_i(\mathbf{Z})$$

- In this case, the WRS statistic can be expressed in the form (3.53) if we choose the coefficients to be the following

$$c_{iN} = \begin{cases} 1 & \text{if } i \leq n \\ 0 & \text{if } i > n \end{cases}$$

and we choose the scores to be

$$a_N(i) = i$$

3.5.2 Properties of Linear Rank Statistics

- The expected value of the linear rank statistic (if the distribution of the Z_i is continuous) is

$$E(T_N) = N\bar{c}_N\bar{a}_N, \tag{3.54}$$

where $\bar{c}_N = \frac{1}{N}\sum_{j=1}^N c_{jN}$ and $\bar{a}_N = \frac{1}{N}\sum_{j=1}^N a_N(j)$

- The formula (3.54) for the expectation only uses the fact that $R_i(\mathbf{Z})$ has a discrete uniform distribution. So,

$$E\{a_N(R_i(\mathbf{Z})\} = \sum_{j=1}^N a_N(j)P\{R_i(\mathbf{Z}) = j\} = \sum_{j=1}^N \frac{a_N(j)}{N} = \bar{a}_N$$

Using this, we can then see that

$$E(T_N) = \sum_{j=1}^N c_{jN}E\{a_N(R_i(\mathbf{Z}))\} = \sum_{j=1}^N c_{jN}\bar{a}_N = N\bar{c}_N\bar{a}_N$$

- A similar argument can show that the variance of T_N is

$$\text{Var}(T_N) = \frac{N^2}{n-1}\sigma_a^2\sigma_c^2,$$

where $\sigma_c^2 = \frac{1}{N}\sum_{j=1}^N (c_{jN} - \bar{c}_N)^2$ and $\sigma_a^2 = \frac{1}{N}\sum_{j=1}^N (a_N(j) - \bar{a}_N)^2$

- To perform hypothesis testing when using a general linear rank statistics, working with the exact distribution or performing permutation tests can often be computationally demanding.
- Using a large-sample approximation is often easier.

- As long as a few conditions for the coefficients and scores are satisfied, one can state the following

$$\frac{T_N - E(T_N)}{\sqrt{\mathrm{Var}(T_N)}} \longrightarrow \mathrm{Normal}(0,1),$$

where, as we showed, both $E(T_N)$ and $\mathrm{Var}(T_N)$ both have closed-form expressions for an arbitrary linear rank statistic.

3.5.3 Other Examples of Linear Rank Statistics

3.5.3.1 The van der Waerden statistic and the normal scores test

- Van der Waerden's rank statistic is used for two-sample problems where the first n observations come from group 1 while the last m observations come from group 2.

- Van der Waerden's rank statistic VW_N is defined as

$$VW_N = \sum_{j=1}^{n} \Phi^{-1}\left(\frac{\mathbf{R}_i(\mathbf{Z})}{N+1} \right)$$

- The function Φ^{-1} denotes the inverse of the cumulative distribution function of a standard Normal random variable.

- The statistic VW_N is a linear rank statistic with coefficients

$$c_{iN} = \begin{cases} 1 & \text{if } i \leq n \\ 0 & \text{if } i > n \end{cases}$$

and scores determined by

$$a_N(i) = \Phi^{-1}\Big(\frac{i}{N+1} \Big)$$

- A test based on van der Waerden's statistic is often referred to as the **normal scores test**.

- The normal scores test is often suggested as an attractive test when the underlying data has an approximately normal distribution.

- If you plot a histogram of the van der Waerden scores $a_N(i)$ it should look roughly like a Gaussian distribution (if there are not too many ties).

3.5.3.2 The median test

- The median test is also a two-sample rank test.
- While the Wilcoxon rank sum test looks at the average rank within group 1, the median test instead looks at how many of the ranks from group 1 are less than the median rank (which should equal $(N+1)/2$).
- The test statistic M_N for the median test is defined as
$$M_N = \sum_{i=1}^{n} I\Big(R_i(\mathbf{Z}) \leq \frac{N+1}{2} \Big)$$
because $(N+1)/2$ will be the median rank.
- This is a linear rank statistic with coefficients
$$c_{iN} = \begin{cases} 1 & \text{if } i \leq n \\ 0 & \text{if } i > n \end{cases}$$
and scores
$$a_N(i) = \begin{cases} 1 & \text{if } i \leq (N+1)/2 \\ 0 & \text{if } i > (N+1)/2 \end{cases}$$
- The median test could be used to test whether or not observations from group 1 tend to be smaller than those from group 2.

3.5.4 Choosing the scores $a_N(i)$

- The rank tests we have discussed so far are nonparametric in the sense that their null distribution does not depend on any particular parametric assumptions about the distributions from which the observations arise.
- For power calculations, we often think of some parameter or "effect size" modifying the base distribution in some way.
- For example, we often think of the shift alternative $F_X(t) = F_Y(t-\theta)$ in the two-sample problem.

- In parametric statistics, when testing $H_0 : \theta = 0$ the most powerful test of $H_0 : \theta = \theta_0$ vs. $H_A : \theta = \theta_A$ is based on rejecting H_0 whenever the likelihood ratio is large enough:
$$\text{Reject } H_0 \text{ if: } \quad \frac{p_{\theta_A}(\mathbf{z})}{p_{\theta_0}(\mathbf{z})} \geq c_{\alpha,n} \tag{3.55}$$
This is the Neyman-Pearson Lemma.

- The same property is true if we are considering tests based on ranks. The most powerful test for testing $H_0 : \theta = \theta_0$ vs. $H_A : \theta = \theta_A$ is based on

$$\text{Reject } H_0 \text{ if: } \quad \frac{P_{\theta_A}\Big(R_1(\mathbf{Z}), \ldots, R_N(\mathbf{Z})\Big)}{P_{\theta_0}\Big(R_1(\mathbf{Z}), \ldots, R_N(\mathbf{Z})\Big)} \geq c_{\alpha,n} \tag{3.56}$$

- The main difference between (3.55) and (3.55) is that the distribution $P_{\theta_A}\Big(R_1(\mathbf{Z}), \ldots, R_N(\mathbf{Z})\Big)$ is unknown unless we are willing to make certain distributional assumptions.
- Nevertheless, we can approximate this probability if θ_A is a location parameter "close" to θ_0

$$P_{\theta_A}\Big(R_1(\mathbf{Z}), \ldots, R_N(\mathbf{Z})\Big) \approx P_{\theta_0}\Big(R_1(\mathbf{Z}), \ldots, R_N(\mathbf{Z})\Big) + \frac{\theta_A}{N!}\sum_{i=1}^{N} c_{iN} E\bigg\{\frac{\partial \log f(Z_{(i)})}{\partial Z}\bigg\}$$

where $Z_{(i)}$ denotes the i^{th} order statistic. See, for example, Chapter 13 of Van der Vaart (2000) for more details on the derivation of this approximation.

- So, large values of the linear rank statistic $T_N = \sum_{i=1}^{N} c_{iN} a_N(i)$ will approximately correspond to large values of $P_{\theta_A}\Big(R_1(\mathbf{Z}), \ldots, R_N(\mathbf{Z})\Big)$ if we choose the scores to be

$$a_N(i) = E\bigg\{\frac{\partial \log f(Z_{(i)})}{\partial Z}\bigg\}$$

- Linear rank statistics with scores generated this way are usually called **locally most powerful** rank tests.

- The best choice of the scores will depend on what we assume about the density f.
- For example, if we assume that $f(z)$ is Normal$(0, 1)$, then

$$\frac{\partial \log f(z)}{\partial z} = -z$$

- The approximate expectation of the order statistics from a Normal$(0, 1)$ distribution are

$$E\{Z_{(i)}\} \approx \Phi^{-1}\bigg(\frac{i}{N+1}\bigg)$$

  This implies that the van der Waerden's scores are approximately optimal if we assume the distribution of the Z_i is Normal.

- This can also be worked out for other choices of $f(z)$.
- If $f(z)$ is a Logistic distribution, the optimal scores correspond to the Wilcoxon rank sum test statistic.
- If $f(z)$ is Laplace (meaning that $f(z) = \frac{1}{2}e^{-|z|}$), then the optimal scores correspond to the median test.

3.6 Additional Reading

- Additional reading which covers the material discussed in this chapter includes:
  - Chapters 3-4 from Hollander et al. (2013)

3.7 Exercises

Exercise 3.1: Suppose X_1, X_2, X_3 are i.i.d. observations from a continuous distribution function F_X. Compute the covariance matrix of the vector of ranks $(R_1(\mathbf{X}), R_2(\mathbf{X}), R_3(\mathbf{X}))$.

Exercise 3.2: Again, suppose that X_1, X_2, X_3, X_4 are i.i.d. observations from a continuous distribution function F_X. Let $T = R_1(\mathbf{X}) + R_2(\mathbf{X})$. Compute $P(T = j)$ for $j = 3, 4, 5, 6, 7$.

Exercise 3.3. Using the exact distribution of the WRS test statistic, what is the smallest possible one-sided p-value associated with the WRS test for a fixed value of n and m (assuming the probability of ties is zero)?

Exercise 3.4. Suppose we have observations $(-2, 1, -1/2, 3/2, 3)$ from a single group. What is the value of the Wilcoxon signed rank statistic?

Exercise 3.5. Under the assumptions of model (3.15), what is the density function of $|D_i|$ and $-|D_i|$?

Exercise 3.6. Under the assumptions of model (3.15) and assuming that $\theta = 0$, show that the expectation of the Wilcoxon signed-rank statistic is 0.

Exercise 3.7: Derive the formula for $c_{\alpha,n}$ shown in (3.24).

Exercise 3.8: Suppose that $X_1, \ldots, X_n \sim$ Exponential(λ) and $Y_1, \ldots, Y_m \sim$ Gamma(α, θ) meaning that the p.d.f. of Y_i is

$$f_{Y_i}(y) = \begin{cases} \frac{\theta^\alpha}{\Gamma(\alpha)} y^{\alpha-1} e^{-\theta y} & \text{if } y > 0 \\ 0 & \text{otherwise.} \end{cases}$$

The pooled-data vector is $\mathbf{Z} = (Z_1, \ldots, Z_{n+m}) = (X_1, \ldots, X_n, Y_1, \ldots, Y_m)$ so that $Z_j = X_j$ for $1 \leq j \leq n$ and $Z_j = Y_{j-n}$ for $n + 1 \leq j \leq n + m$.

Compute both $E\{R_1(\mathbf{Z})\}$ and $E\{R_{n+1}(\mathbf{Z})\}$.

Chapter 4

Rank Tests for Multiple Groups

- We can roughly think of the tests discussed in Chapter 3 as being related to the well-known parametric tests shown in the table below.

Parametric Test	**Nonparametric Tests**
One-Sample t-test	Wilcoxon Signed Rank/Sign Test
Two-Sample t-test	Wilcoxon Rank Sum/Normal Scores/Median Test

- The **Kruskal-Wallis** test can be though of as the nonparametric analogue of one-way analysis of variance (ANOVA).

- For $K \geq 3$ groups, one-way ANOVA considers the analysis of observations arising from the following model

$$Y_{kj} = \mu_k + \varepsilon_{kj}, \qquad j = 1, \dots, n_k; k = 1, \dots, K \tag{4.1}$$

  where it is often assumed that $\varepsilon_{kj} \sim \text{Normal}(0, \sigma^2)$.

- Usually, the one-way ANOVA hypothesis of interest is something like

$$H_0 : \mu_1 = \mu_2 = \dots = \mu_K \tag{4.2}$$

  which is often referred to as the homogeneity hypothesis.

- A test of the hypothesis (4.2) is based on decomposing the observed variation in the responses Y_{kj}:

$$\begin{aligned}
\underbrace{\sum_{k=1}^{K}\sum_{j=1}^{n_k}(Y_{kj} - \bar{Y}_{..})^2}_{SST} &= \sum_{k=1}^{K}\sum_{j=1}^{n_k}(\bar{Y}_{k.} - \bar{Y}_{..})^2 + \sum_{k=1}^{K}\sum_{j=1}^{n_k}(Y_{kj} - \bar{Y}_{k.})^2 \\
&= \underbrace{\sum_{k=1}^{K} n_k(\bar{Y}_{k.} - \bar{Y}_{..})^2}_{SSA} + \underbrace{\sum_{k=1}^{K}\sum_{j=1}^{n_k}(Y_{kj} - \bar{Y}_{k.})^2}_{SSE} \quad (4.3)
\end{aligned}$$

where $\bar{Y}_{k.} = \frac{1}{n_k}\sum_{j=1}^{n_k} Y_{kj}$ and $\bar{Y}_{..} = \frac{1}{K}\sum_{k=1}^{K} \bar{Y}_{k.}$.

- Large values of $SSA = \sum_{k=1}^{K} n_k(\bar{Y}_{k.} - \bar{Y}_{..})^2$ provide evidence against the null hypothesis (4.2). The alternative hypothesis here is that there is at least one pair of means μ_h, μ_l such that $\mu_h \neq \mu_l$.

4.1 The Kruskal-Wallis Test

4.1.1 Definition

- Instead of assuming (4.1) for the responses Y_{kj}, a nonparametric way of thinking about this problem is to instead only assume that

$$Y_{kj} \sim F_k \quad (4.4)$$

That is, $Y_{k1}, Y_{k2}, \dots, Y_{kn_k}$ is an i.i.d. sample from F_k for each k.

- A nonparametric version of the one-way ANOVA homogeneity hypothesis is

$$H_0 : F_1 = F_2 = \dots = F_K \quad (4.5)$$

- The "shift alternative" in this case can be stated as

$$H_A : F_k(t) = F(t - \Delta_k), \quad \text{for } k = 1, \dots K \quad \text{and not all } \Delta_k \text{ equal} \quad (4.6)$$

- The **Kruskal-Wallis test statistic** KW_N is similar to the SSA term (defined in (4.3)) in the one-way ANOVA setting.

- Rather than comparing the group-specific means $\bar{Y}_{k.}$ with the overall mean $\bar{Y}_{..}$, the Kruskal-Wallis test statistic compares the group-specific rank means $\bar{R}_{k.}$ with their overall expectation under the null hypothesis.

- The Kruskal-Wallis test statistic is defined as

$$KW_N = \frac{12}{N(N-1)} \sum_{k=1}^{K} n_k \Big(\bar{R}_{k.} - \frac{N+1}{2} \Big)^2, \quad \text{where } N = \sum_{k=1}^{K} n_k \tag{4.7}$$

- In (4.7), $\bar{R}_{k.}$ is the average rank of those in the k^{th} group

$$\bar{R}_{k.} = \frac{1}{n_k} \sum_{j=1}^{n_k} R_{kj}(\mathbf{Z}), \tag{4.8}$$

  where $\mathbf{Z}$ denotes the pooled-data vector and $R_{kj}(\mathbf{Z})$ denotes the rank of Y_{kj} in the "pooled-data ranking".

- What is the expectation of $\bar{R}_{k.}$ under the null hypothesis (4.5)?
- Again, if the null hypothesis is true, we can treat all of our responses Y_{kj} as just an i.i.d. sample of size N from a common distribution function F.
- Hence, as we showed in (3.5) from Chapter 3, $E\{R_{kj}(\mathbf{Z})\} = (N+1)/2$ under the assumption that the data are an i.i.d. sample from a common distribution function.
- So, the intuition behind the definition of KW_N is that the differences $\bar{R}_{k.} - \frac{N+1}{2}$ should be small whenever the homogeneity hypothesis (4.5) is true.

- When $K = 2$, the following relationship between the Kruskal-Wallis statistic KW_N and the Wilcoxon rank sum test statistic W from Chapter 3 holds.

$$KW_N = \frac{12}{mn(N+1)} \Big(W - \frac{n(N+1)}{2} \Big)^2. \tag{4.9}$$

- Hence, the p-value from a Kruskal-Wallis test and a (two-sided) WRS test should be the same when $K = 2$.
- However, you cannot directly perform a one-sided test using the Kruskal-Wallis test.

- **Exercise 4.1** If $K = 2$, show that equation (4.9) holds.

Group	Y	Rank
Group 1	1.00	8
Group 1	-1.20	2
Group 1	-1.50	1
Group 2	0.00	5
Group 2	-0.10	4
Group 2	1.10	9
Group 3	0.90	7
Group 3	-0.40	3
Group 3	0.60	6

An Example

- Consider the data shown in the table. In this case, $N = 9$, $\bar{R}_{1.} = 11/3$, $\bar{R}_{2.} = 6$, and $\bar{R}_{3.} = 16/3$. The Kruskall-Wallis statistic is

$$KW_N = \frac{1}{2}\Big\{3(11/3 - 5)^2 + 3(6 - 5)^2 + 3(16/3 - 5)^2\Big\} = 13/9$$

4.1.2 Asymptotic Distribution and Connection to One-Way ANOVA

- The Kruskal-Wallis statistic KW_N has an asymptotic chi-square distribution with $K - 1$ degrees of freedom under the null hypothesis (4.5).
- This follows from the fact that $(\bar{R}_{k.} - (N+1)/2)$ is approximately normally distributed for large n_k.
- **R** uses the large-sample approximation when computing the p-value for the Kruskal-Wallis test.

- The Kruskal-Wallis test can also be thought of as the test you would obtain if you applied the one-way ANOVA setup to the ranks of Y_{kj}.
- The one-way ANOVA test is based on the value of SSA where, as in (4.3), SSA is defined as

$$SSA = \sum_{k=1}^{K} n_k(\bar{Y}_{k.} - \bar{Y}_{..})^2$$

- You then reject H_0, when $SSA/SSE = SSA/(SST - SSA)$ is sufficiently large.

- Notice that if we computed SSA using the ranks $R_{kj}(\mathbf{Z})$ rather than the observations Y_{kj}, we would get:

$$\begin{aligned} SSA_r &= \sum_{k=1}^{K} n_k \Big(\bar{R}_{k.} - \bar{R}_{..} \Big)^2 \\ &= \sum_{k=1}^{K} n_k (\bar{R}_{k.} - \frac{N+1}{2})^2 \\ &= \frac{N(N-1)}{12} KW_N \end{aligned} \tag{4.10}$$

- If you were applying ANOVA to the ranks of Y_{kj}, SST_r would be the following fixed constant:

$$\text{SST}_r = \frac{N(N+1)(N-1)}{12}$$

- So, any test of the homogeneity hypothesis would be based on just the value of SSA_r which, as we showed in (4.10), is just a constant times the Kruskal-Wallis statistic.

4.2 Performing the Kruskal-Wallis Test in R

- We will look at performing Kruskal-Wallis tests in **R** by using the "InsectSprays" dataset.

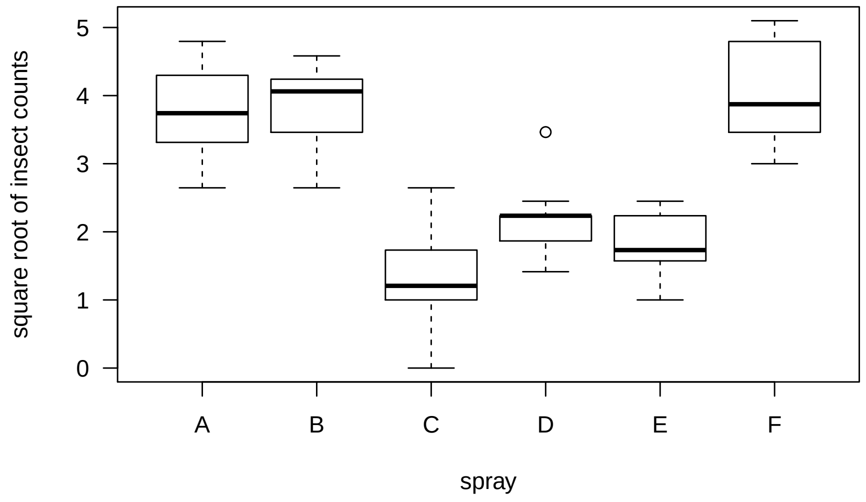
```
head(InsectSprays)
```

```
##   count spray
## 1    10     A
## 2     7     A
## 3    20     A
## 4    14     A
## 5    14     A
## 6    12     A
```

- This dataset has 72 observations.
- The variable **count** is the number of insects measured in some agricultural unit.
- The variable **spray** was the type of spray used on that unit.

- You could certainly argue that a standard ANOVA is not appropriate in this situation because the responses are counts, and for count data, the variance is usually a function of the mean.
- A generalized linear model with a log link function might be more appropriate.
- Applying a square-root transformation to count data is also a commonly suggested approach. (The square-root transformation is the "variance-stabilizing transformation" for Poisson-distributed data).

```
boxplot(sqrt(count) ~ spray, data=InsectSprays,las=1,
        ylab="square root of insect counts")
```



- Let us perform a test of homogeneity using both the one-way ANOVA approach and a Kruskal-Wallis test

```
anova(lm(sqrt(count) ~ spray, data=InsectSprays))
```

```
## Analysis of Variance Table
##
```

```
## Response: sqrt(count)
##           Df Sum Sq Mean Sq F value    Pr(>F)
## spray      5 88.438 17.6876  44.799 < 2.2e-16 ***
## Residuals 66 26.058  0.3948
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
a  <- kruskal.test(sqrt(count) ~ spray, data=InsectSprays)
a$p.value
```

```
## [1] 1.510844e-10
```

- Notice that applying the square root transformation does not affect the value of the Kruskal-Wallis statistic or the Kruskal-Wallis p-value.

```
kruskal.test(count ~ spray, data=InsectSprays)
```

```
##
##  Kruskal-Wallis rank sum test
##
## data:  count by spray
## Kruskal-Wallis chi-squared = 54.691, df = 5, p-value = 1.511e-10
```

- This invariance to data transformation is not true for the standard one-way ANOVA.

```
anova(lm(count ~ spray, data=InsectSprays))
```

```
## Analysis of Variance Table
##
## Response: count
##           Df Sum Sq Mean Sq F value    Pr(>F)
## spray      5 2668.8  533.77  34.702 < 2.2e-16 ***
## Residuals 66 1015.2   15.38
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

4.3 Comparison of Specific Groups

- A Kruskal-Wallis test performs a test of the overall homogeneity hypothesis

$$H_0 : F_1 = F_2 = ... = F_K \tag{4.11}$$

- However, a rejection of the homogeneity hypothesis does not indicate which group differences are primarily the source of this rejection nor does it provide any measure of the "magnitude" of the differences between each of the groups.

- Dunn's test is the suggested way to compute pairwise tests of stochatic dominance.

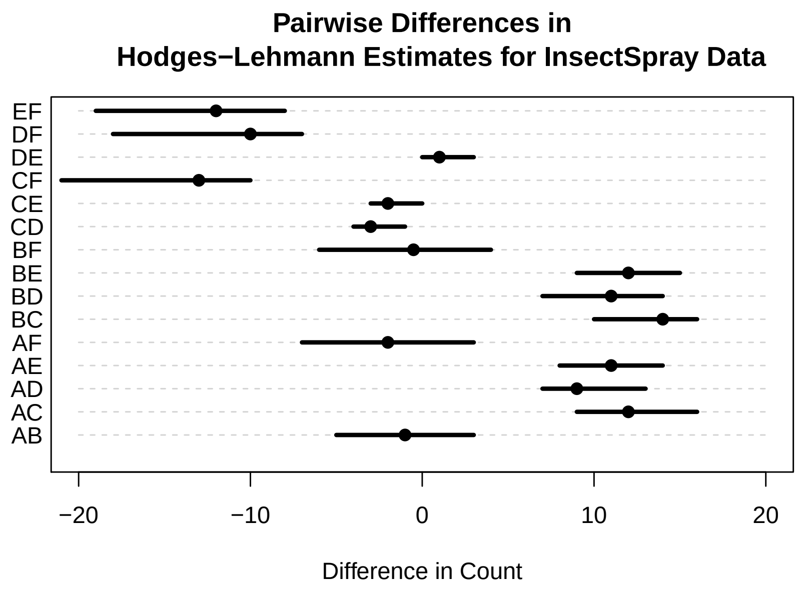
- Performing a series of pairwise Wilcoxon rank sum test can lead to violations of transitivity. For example, group A is "better" than B which is better than C, but group C is better than A.

- In **R**, Dunn's test can be performed using the **dunn.test** package.

- In traditional one-way ANOVA one often reports pairwise differences in the means and their associated confidence intervals.

- In the context of a Kruskal-Wallis test, one could report pairwise differences in the Hodges-Lehmann estimate though other comparisons may also be of interest.

- One nice approach is to use the proportional odds model interpretation of the Kruskal-Wallis test and then report the difference in the estimated proportional odds coefficients. See Section 7.6 of http://hbiostat.org/doc/bbr.pdf for more details on the proportional odds model.



4.4 An Additional Example

- We will use the “cane” dataset from the **boot** package.

```
library(boot)
data(cane)
head(cane)
```

```
##     n  r  x var block
## 1  87 76 19   1     A
## 2 119  8 14   2     A
## 3  94 74  9   3     A
## 4  95 11 12   4     A
## 5 134  0 12   5     A
## 6  92  0  3   6     A
```

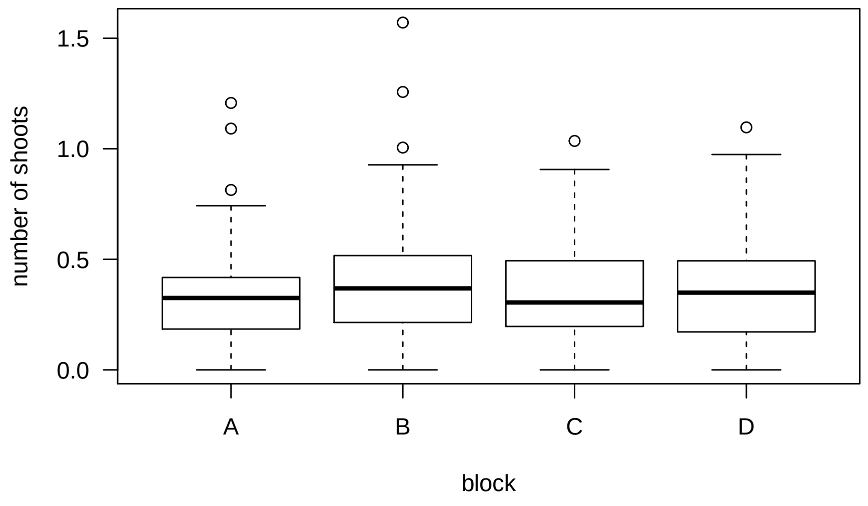
- These data come from a study trying to determine the susceptibility of different types of sugar cane to a particular type of disease.
- The variable **n** contains the total number of shoots in each plot.
- The variable **r** containes the total number of diseased shoots.

- We can create a new variable **prop** that measures the proportion of shoots that are diseased.

```
cane$prop <- cane$r/cane$n
```

- You could certainly argue that a logistic regression model is a better approach here, but we will analyze the transformed proportions using the arcsine square root transformation.

```
cane$prop.trans <- asin(sqrt(cane$prop))
boxplot(prop.trans ~ block, data=cane, las=1, ylab="number of shoots")
```



```
kruskal.test(prop ~ block, data=cane)
```

```
##
##  Kruskal-Wallis rank sum test
##
## data:  prop by block
## Kruskal-Wallis chi-squared = 1.1355, df = 3, p-value = 0.7685
```

4.5 Additional Reading

- Additional reading which covers the material discussed in this chapter includes:
  - Chapters 6 from Hollander et al. (2013)

Chapter 5

Permutation Tests

- Permutation tests are a useful tool that allows you to avoid depending on specific parametric assumptions.
- Permutation tests are also useful in more complex modern applications where it can be difficult to work out the theoretical null distribution of the desired test statistic.

5.1 Notation

- A **permutation** π of a set S is a function $\pi : S \longrightarrow S$ is a function that is both one-to-one and onto.
- We will usually think of S as the set of observation indices in which case $S = \{1, \ldots, N\}$ for sample size N.
- Each permutation π of $S = \{1, \ldots, N\}$ defines a particular ordering of the elements of S. For this reason, a permutation is often expressed as the following ordered list

$$\pi = (\pi(1), \pi(2), \ldots, \pi(N))$$

- In other words, we can think of a permutation of S as a particular ordering of the elements of S.
- For example, if $S = \{1, 2, 3\}$, and π_1 is a permutation of S defined as $\pi_1(1) = 3$, $\pi_1(2) = 1$, $\pi_1(3) = 2$, then this permutation expressed as an ordered list would be

$$\pi_1 = (3, 1, 2)$$

- There are 5 other possible permutations of S:

$$\begin{aligned}
\pi_2 &= (1, 2, 3) \\
\pi_3 &= (2, 1, 3) \\
\pi_4 &= (1, 3, 2) \\
\pi_5 &= (3, 2, 1) \\
\pi_6 &= (2, 3, 1)
\end{aligned}$$

- If S has N distinct elements, there are $N!$ possible permutations of S.
- We will let $\mathcal{S}_N$ denote the set of all permutations of the set $\{1, \dots, N\}$.

5.2 Permutation Tests for the Two-Sample Problem

- Permutation tests for two-sample problems are motivated by the following reasoning:
  - If there is no real difference between the two groups, there is nothing "special" about the difference in means between the two groups.
  - The observed difference in the mean between the two groups should not be notably different than mean differences from randomly formed groups.
  - Forming "random" groups can be done by using many permutations of the original data.

5.2.1 Example 1

- Suppose we have observations from two groups $X_1, \dots, X_n \sim F_X$ and $Y_1, \dots, Y_m \sim F_Y$.
- Let $\mathbf{Z} = (Z_1, \dots, Z_N)$ denote the pooled data

$$(Z_1, \dots, Z_N) = (X_1, \dots, X_n, Y_1, \dots, Y_m) \tag{5.1}$$

- For a permutation π of $\{1, \dots, N\}$, we will let $\mathbf{Z}_\pi$ denote the corresponding permuted dataset

$$\mathbf{Z}_\pi = (Z_{\pi(1)}, Z_{\pi(2)}, \dots, Z_{\pi(N)}) \tag{5.2}$$

- For example, the columns in Table 5.1 are just permutations of the original data $\mathbf{Z}$.

	OriginalData	Perm1	Perm2	Perm3	Perm4	Perm5
z1	0.60	-0.60	0.60	-0.90	0.70	0.60
z2	-0.80	-1.40	-0.60	0.70	-0.40	-0.60
z3	-0.60	0.70	0.20	0.60	-1.40	-0.80
z4	-0.90	0.20	-0.40	0.20	0.20	0.30
z5	0.30	-0.40	-1.30	-0.40	-0.90	-0.40
z6	-1.30	-1.30	-1.40	-0.60	-0.80	0.70
z7	0.20	0.30	0.70	-1.40	0.30	-0.90
z8	0.70	0.60	0.30	-1.30	0.60	0.20
z9	-1.40	-0.90	-0.80	0.30	-0.60	-1.40
z10	-0.40	-0.80	-0.90	-0.80	-1.30	-1.30
mean difference	0.16	0.12	0.12	0.80	0.00	0.36

Table 5.1: Example of Permuting a Vector of Responses. This example assumes n=m=5.

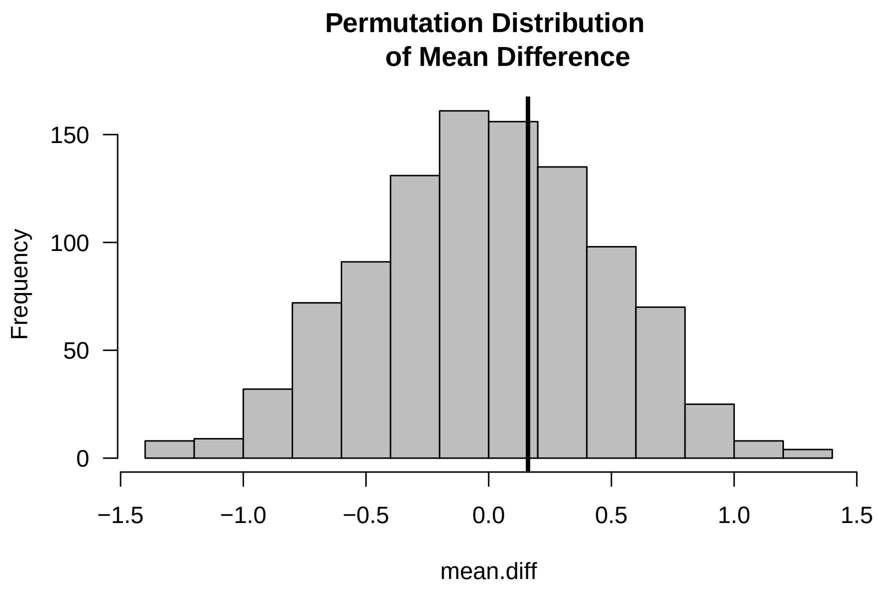
- Suppose we want to base a test on the difference in the means between the two groups

$$T_N(\mathbf{Z}) = \bar{X} - \bar{Y} = \frac{1}{n}\sum_{i=1}^{n} Z_i - \frac{1}{m}\sum_{i=n+1}^{N} Z_i \tag{5.3}$$

- We will let t_{obs} denote the observed value of the mean difference. That is, $t_{obs} = T_N(\mathbf{Z}_{obs})$, where $\mathbf{Z}_{obs}$ is the vector of the observed data.

- Under the null hypothesis that $F_X = F_Y$, the observed mean difference should not be "abnormal" when compared with the mean differences from many other permutations of the data.

```r
z <- c(0.6, -0.8, -0.6, -0.9, 0.3, -1.3, 0.2, 0.7, -1.4, -0.4) ## data
observed.diff <- mean(z[1:5]) - mean(z[6:10])  ## observed mean difference
nperms <- 1000
mean.diff <- rep(NA, nperms)
for(k in 1:nperms) {
    ss <- sample(1:10, size=10)  ## draw a random permutation
    z.perm <- z[ss]   ## form the permuted dataset
    mean.diff[k] <- mean(z.perm[1:5]) - mean(z.perm[6:10]) ## compute mean difference
                                                            ## for permuted dataset
}
hist(mean.diff, las=1, col="grey", main="Permutation Distribution
     of Mean Difference")
abline(v=observed.diff, lwd=3)
```



5.2.2 Permutation Test p-values

- The one-sided p-value for the permutation test is

$$\begin{aligned}\text{p-value} &= \frac{\text{number of permutations such that } T_N \geq t_{obs}}{N!} \\ &= \frac{1}{N!}\sum_{\pi \in \mathcal{S}_N} I\Big(T_N(\mathbf{Z}_\pi) \geq t_{obs}\Big)\end{aligned}$$

- The two-sided p-value for the two-sample problem would be

$$\text{p-value} = \frac{1}{N!}\sum_{\pi \in \mathcal{S}_N} I\Big(\Big|T_N(\mathbf{Z}_\pi)\Big| \geq |t_{obs}|\Big)$$

- As we did when producing the above histogram, the permutation-test p-value is often computed by using a large number of random permutations rather than computing the test statistic for every possible permutation.

- The Monte Carlo permutation-test p-value is defined as

$$\text{p-value}_{mc} = \frac{1}{S+1}\left[1 + \sum_{s=1}^{S} I\Big(T_N(\mathbf{Z}_{\pi_s}) \geq t_{obs}\Big)\right] \tag{5.4}$$

  where $\pi_1, \ldots, \pi_S$ are randomly drawn permutations

- The two-sided (Monte Carlo) p-value for the example shown in Table 5.1 is

```
pval.mc <- (1 + sum(abs(mean.diff) >= abs(observed.diff)))/(nperms + 1)
round(pval.mc, 2)
```

```
## [1] 0.77
```

5.2.3 Example 2: Ratios of Means

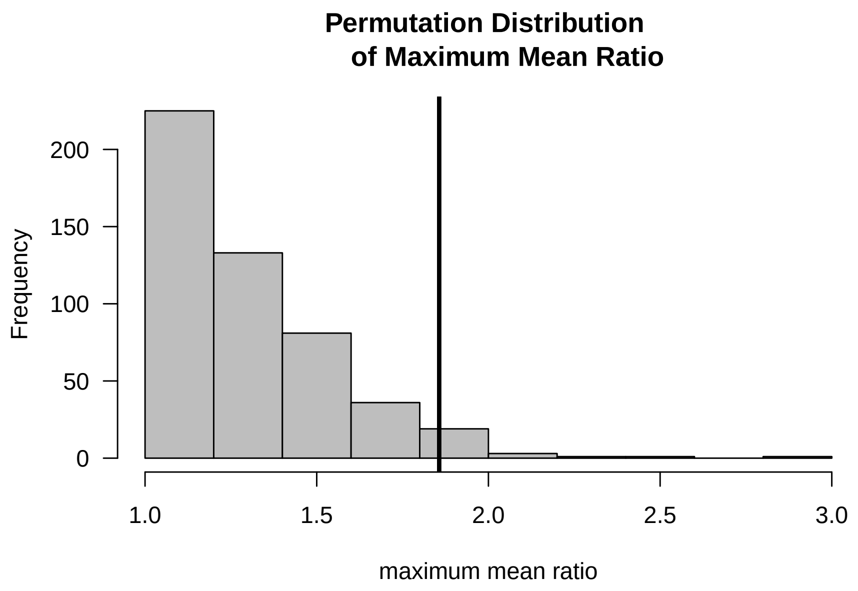
- With permutation tests, you are not limited to difference in means. You can choose the statistic $T_N(\mathbf{Z})$ to measure other contrasts of interest.

- For example, with nonnegative data you might be interested in the ratio of means between the two groups

$$T_N(\mathbf{Z}) = \max\left\{\bar{X}/\bar{Y}, \bar{Y}/\bar{X}\right\} \tag{5.5}$$

- Let us see how this works for a simulated example with $n = m = 20$ where we assume that $X_1, \ldots, X_n \sim$ Exponential(1) and $Y_1, \ldots, Y_m \sim$ Exponential(1/2).

```
set.seed(5127)
xx <- rexp(20, rate=1)
yy <- rexp(20, rate=0.5)
zz <- c(xx, yy) ## this is the original data
t.obs <- max(mean(zz[1:20])/mean(zz[21:40]), mean(zz[21:40])/mean(zz[1:20]))
nperms <- 500
mean.ratio <- rep(0, nperms)
for(k in 1:nperms) {
     ss <- sample(1:40, size=40)
     zz.perm <- zz[ss]
     mean.ratio[k] <- max(mean(zz.perm[1:20])/mean(zz.perm[21:40]),
                          mean(zz.perm[21:40])/mean(zz.perm[1:20]))
}
hist(mean.ratio, las=1, col="grey", main="Permutation Distribution
     of Maximum Mean Ratio", xlab="maximum mean ratio")
abline(v=t.obs, lwd=3)
```



- The two-side (Monte Carlo) permutation test p-value is:

```
pval.mc <- (1 + sum(mean.ratio >= t.obs))/(nperms + 1)
round(pval.mc, 2)
```

```
## [1] 0.04
```

5.2.4 Example 3: Differences in Quantiles

- Permutation tests are especially useful in problems where working out the null distribution is difficult, or when certain approximations of the null distributions are hard to justify.
- An example of this occurrs if you want to compare medians, or more generally, compare quantiles between two groups.
- The difference-in-quantiles statistic would be defined as

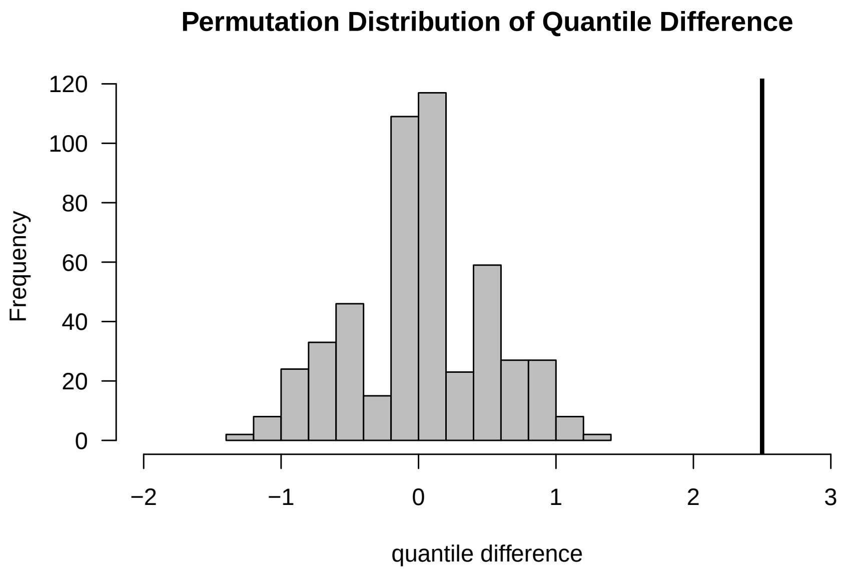
$$T_N(\mathbf{Z}) = Q_p(Z_1, \dots, Z_n) - Q_p(Z_{n+1}, \dots, Z_N)$$

  where $Q_p(X_1, \dots, X_H)$ denotes the p^{th} quantile from the data $X_1, \dots, X_H$.
- The difference in quantiles could be computed with the following **R** code:

```r
z <- rnorm(10)
quantile(z[1:5], probs=.3) - quantile(z[6:10], probs=.3)
```

```
##       30%
## 0.2671133
```

- Note that setting **probs=.5** in the **quantile** function will return the median.



- **Exercise 5.1** Suppose we have the following data from two groups $(X_1, X_2, X_3) = (-1, 0, 1)$ and $(Y_1, Y_2, Y_3) = (4, -2, 2)$. Compute the (two-sided) permutation p-value for the following two statistics:
  - $T_N(\mathbf{Z}) = \text{median}(X_1, X_2, X_3) - \text{median}(Y_1, Y_2, Y_3)$.
  - $T_N(\mathbf{Z}) = \bar{X} - \bar{Y}$.
- **Exercise 5.2.** Suppose we have data from two groups such that $X_1, \ldots, X_n \sim \text{Normal}(0, 1)$ and $Y_1, \ldots, Y_m \sim \text{Normal}(1, 1)$. Using $n = m = 50$ and 500 simulation replications, compute 500 significance thresholds from the one-sided permutation test which uses the statistic $T_N(\mathbf{Z}) = \bar{X} - \bar{Y}$. How, does this compare with the t-statistic threshold of approximately $1.64 * \sqrt{2/50}$?

- **Exercise 5.3.** Suppose we have data from two groups such that $X_1, \dots, X_n \sim \text{Normal}(0, 1)$ and $Y_1, \dots, Y_m \sim \text{Normal}(1, 1)$. Using $n = m = 50$ and 500 simulation replications, compute the power of the permutation test which uses the statistic $T_N(\mathbf{Z}) = \bar{X} - \bar{Y}$ to detect this true alternative. How, does the power compare with the (two-sided) two-sample t-statistic and the (two-sided) Wilcoxon rank sum test?

5.3 The Permutation Test as a Conditional Test

- A permutation test is an example of a **conditional test**.
- Typically, the p-value is defined as

$$\text{p-value} = P(T \geq t_{obs} | H_0) \tag{5.6}$$

  for some test statistic T.
- In other words, the p-value is the probability that a random variable which follows the null distribution exceeds t_{obs}.
- In many problems however, the null hypothesis H_0 is not determined by a single parameter but contains many parameters.
- For example, the null hypothesis in a t-test is really $H_0 : \mu_x = \mu_y$ and $\sigma > 0$. That is, the null hypothesis is true for many different values of σ.

- When H_0 contains many parameter values, one approach for computing a p-value is to choose the test statistic T so that its distribution is the same for every point in H_0.
- A more general approach is to instead compute a **conditional p-value**
- A conditional p-value is defined as

$$\text{p-value} = P(T \geq t_{obs} | S = s, H_0)$$

  where S is a sufficient statistic for the unknown terms in H_0.
- A classic example of this is Fisher's exact test.

- A permutation test computes a conditional p-value where the sufficient statistic is the vector of order statistics $(Z_{(1)}, Z_{(2)}, \dots, Z_{(N)})$.
- Recall that the order statistics are defined as

$$\begin{aligned} Z_{(1)} &= \text{smallest observation} \\ Z_{(2)} &= \text{second smallest observation} \\ &\vdots \\ Z_{(N)} &= \text{largest observation} \end{aligned}$$

- What is the conditional distribution of the observed data conditional on the observed order statistics?
- It is:

$$\begin{aligned} f_{Z_1,\dots,Z_N|Z_{(1)},\dots,Z_{(N)}}(z_{\pi(1)},\dots,z_{\pi(N)}|z_1,\dots,z_N) &= \frac{f_{Z_1,\dots,Z_N}(z_{\pi(1)},\dots,z_{\pi(N)})}{f_{Z_{(1)},\dots,Z_{(N)}}(z_1,\dots,z_N)} \\ &= \frac{f_{Z_i}(z_{\pi(1)})\cdots f_{Z_i}(z_{\pi(N)})}{N! f_{Z_i}(z_1)\cdots f_{Z_i}(z_N)} \\ &= \frac{1}{N!} \end{aligned}$$

(See Chapter 5 of Casella and Berger (2002) for a detailed description of the distribution of order statistics)

- In other words, if we know the value of: $Z_{(1)} = z_1, \dots, Z_{(N)} = z_N$, then any event of the form $\{Z_1 = z_{\pi(1)}, \dots, Z_N = z_{\pi(N)}\}$ has an equal probability of occurring for any permutation chosen.
- This equal probability of $1/N!$ is only true under H_0 where we can regard the data as coming from a common distribution.

- If we are conditioning on $Z_{(1)} = z_1, \dots, Z_{(N)} = z_N$, then the probability that $T_N(Z_1, \dots, Z_N) \geq t$ is just the number of permutations of $(z_1, \dots, z_N)$ where the test statistic is greater than t divided by $N!$.
- In other words

$$P\Big\{T_N(Z_1,\dots,Z_N) \geq t | Z_{(1)} = z_1, \dots, Z_{(N)} = z_N\Big\} \tag{5.7}$$

$$= \frac{1}{N!}\sum_{\pi \in \mathcal{S}_N} I\Big(T_N(z_{\pi(1)},\dots,z_{\pi(N)}) \geq t\Big) \tag{5.8}$$

- Let us now consider a concrete example.

- Suppose we have a two-sample problem with four observations. The first two observations come from the first group while the last two observations come from the second group.

- The order statistics that we will condition on are:

$$\begin{aligned} Z_{(1)} &= z_1 = -3 && (5.9) \\ Z_{(2)} &= z_2 = -1 && (5.10) \\ Z_{(3)} &= z_3 = 2 && (5.11) \\ Z_{(4)} &= z_4 = 5 && (5.12) \end{aligned}$$

- If T_4 is the mean difference

$$T_4(Z_1, Z_2, Z_3, Z_4) = \frac{Z_1 + Z_2 - Z_3 - Z_4}{2} \tag{5.13}$$

  what is the probability

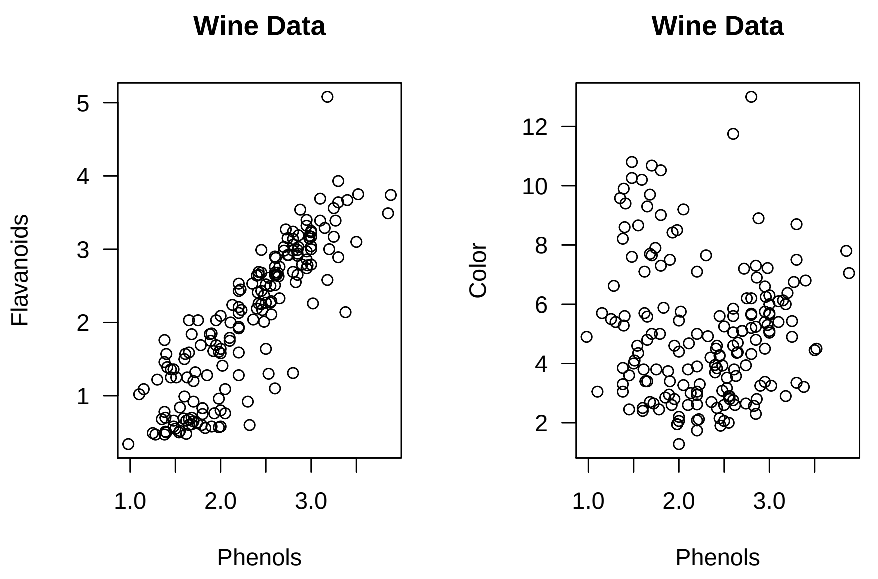
$$P\Big\{T_4(Z_1, Z_2, Z_3, Z_4) \geq 2.5 | Z_{(1)} = z_1, Z_{(2)} = z_2, Z_{(3)} = z_3, Z_{(4)} = z_4\Big\} \tag{5.14}$$

- From the below table, we see that the number of times $T_4 \geq 2.5$ occurs is 8. Hence,

$$\begin{aligned} & P\Big\{T_4(Z_1, Z_2, Z_3, Z_4) \geq 2.5 | Z_{(1)} = z_1, Z_{(2)} = z_2, Z_{(3)} = z_3, Z_{(4)} = z_4\Big\} \\ = \; & 8/24 = 1/3. \end{aligned}$$

a1	a2	a3	a4	P(Z1 = a1, Z2=a2, Z3=a3, Z4=a4|order stat)	T(a1, a2, a3, a4)	T(a1, a2, a3, a4) >= 2.5
-3	-1	2	5	1/24	-5.50	0
-3	-1	5	2	1/24	-5.50	0
-3	2	-1	5	1/24	-2.50	0
-3	2	5	-1	1/24	-2.50	0
-3	5	-1	2	1/24	0.50	0
-3	5	2	-1	1/24	0.50	0
-1	-3	2	5	1/24	-5.50	0
-1	-3	5	2	1/24	-5.50	0
-1	2	-3	5	1/24	-0.50	0
-1	2	5	-3	1/24	-0.50	0
-1	5	-3	2	1/24	2.50	1
-1	5	2	-3	1/24	2.50	1
2	-3	-1	5	1/24	-2.50	0
2	-3	5	-1	1/24	-2.50	0
2	-1	-3	5	1/24	-0.50	0
2	-1	5	-3	1/24	-0.50	0
2	5	-3	-1	1/24	5.50	1
2	5	-1	-3	1/24	5.50	1
5	-3	-1	2	1/24	0.50	0
5	-3	2	-1	1/24	0.50	0
5	-1	-3	2	1/24	2.50	1
5	-1	2	-3	1/24	2.50	1
5	2	-3	-1	1/24	5.50	1
5	2	-1	-3	1/24	5.50	1

5.4 A Permutation Test for Correlation



- Suppose we have N pairs of observations $(U_1, V_1), \dots, (U_N, V_N)$
- The correlation between these pairs is defined as

$$\rho_{UV} = \frac{\mathrm{Cov}(U_i, V_i)}{\sigma_U \sigma_V} \tag{5.15}$$

- The test statistic of interest here is the sample correlation

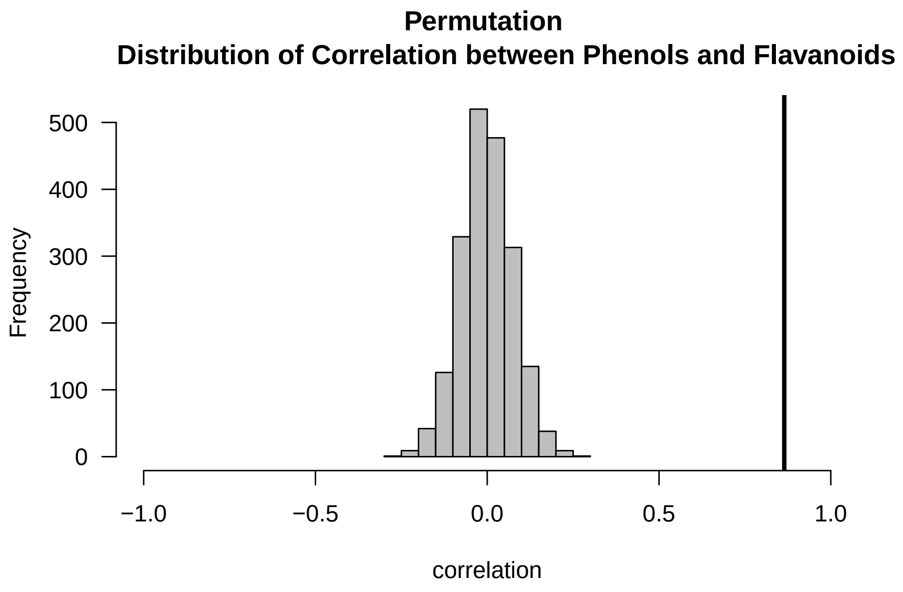
$$T_N(\mathbf{U}, \mathbf{V}) = \frac{\sum_{i=1}^N (U_i - \bar{U})(V_i - \bar{V})}{\sqrt{\sum_{i=1}^N (U_i - \bar{U})^2 \sum_{i=1}^N (V_i - \bar{V})^2}} \tag{5.16}$$

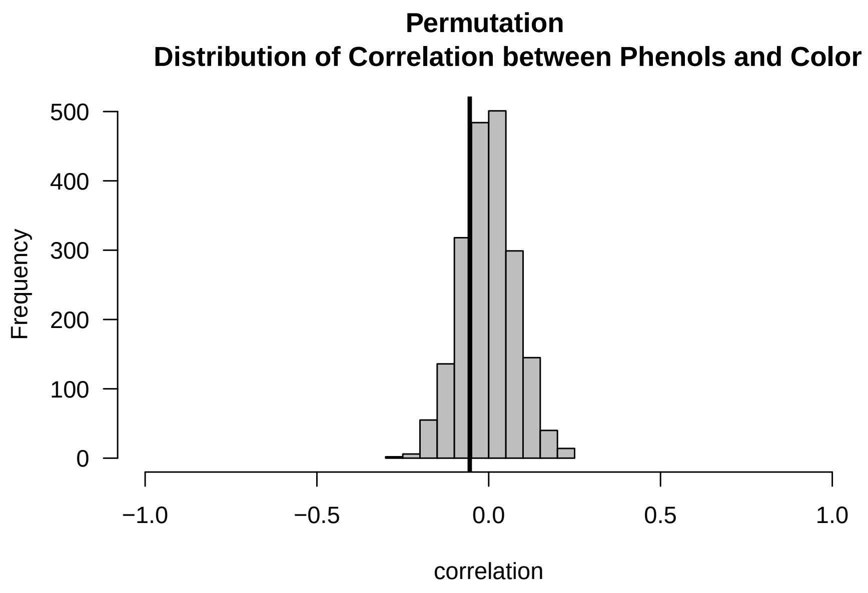
- To find the the permutation distribution, we only need to look at $T_N(\mathbf{U}_\pi, \mathbf{V})$ for different permutations π.
- In other words, we are computing correlation among pairs of the form $(U_{\pi(1)}, V_1), \dots, (U_{\pi(N)}, V_N)$.
- We only need to look at $\mathbf{U}_\pi$ because this achieves the objective of randomly "switching observation pairs".

- The two-sided p-value for the permutation test of $H_0 : \rho_{UV} = 0$ vs. $H_A : \rho_{UV} \neq 0$ is

$$\text{p-value} = \frac{1}{N!} \sum_{\pi \in \mathcal{S}_N} I\Big(\big| T_N(\mathbf{U}_{\pi}, \mathbf{V}) \big| \geq |t_{obs}| \Big)$$

```
library(rattle.data)
## Computing the permutation distribution for correlation
## between flavanoids and phenols

n.obs <- nrow(wine) ## number of observations
t.obs.pf <- cor(wine$Phenols, wine$Flavanoids) ## observed correlation
nperms <- 2000
cor.perm.pf <- rep(0, nperms)
for(k in 1:nperms) {
     ss <- sample(1:n.obs, size=n.obs)
     uu.perm <- wine$Phenols[ss]
     cor.perm.pf[k] <- cor(uu.perm, wine$Flavanoids)
}
hist(cor.perm.pf, xlim=c(-1, 1), las=1, col="grey", main="Permutation
     Distribution of Correlation between Phenols and Flavanoids",
     xlab="correlation")
abline(v=t.obs.pf, lwd=3)
```





- Now let us compute the p-values for both the Phenols/Flavanoids and Phenols/Color association tests.

```r
# p-value for correlation between Phenols/Flavanoids
pval.mc <- (1 + sum(abs(cor.perm.pf) >= abs(t.obs.pf)))/(nperms + 1)
round(pval.mc, 4)
```

```
## [1] 5e-04
```

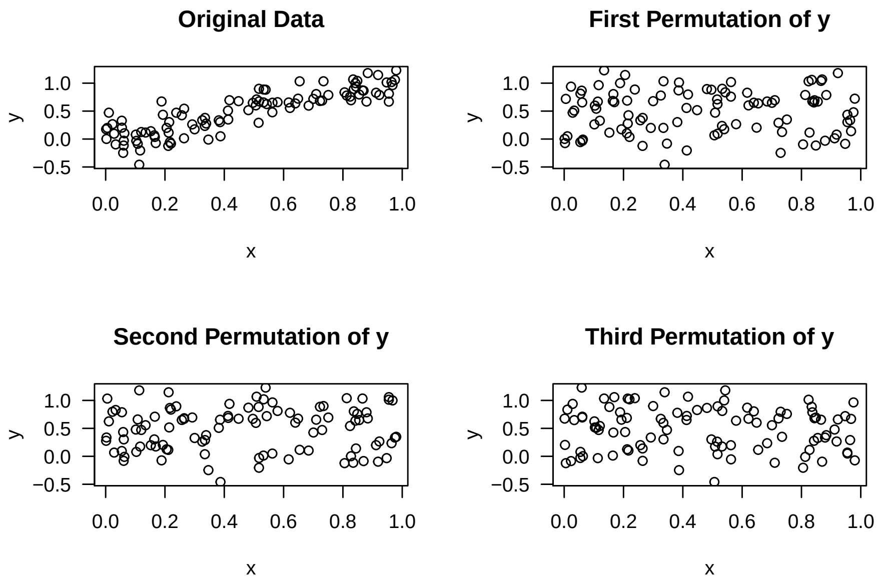
```r
# p-value for correlation between Phenols/Color
pval.mc <- (1 + sum(abs(cor.perm.pc) >= abs(t.obs.pc)))/(nperms + 1)
round(pval.mc, 4)
```

```
## [1] 0.4648
```

5.5 A Permutation Test for Variable Importance in Regression and Machine Learning

- The idea of permutation testing can also be applied in the context of regression.

- In regression, we have a series of responses $y_1, \ldots, y_N$, and we have a series of associated covariates vectors $\mathbf{x}_i$.
- For regression, we are going to perform permutations on the vector of responses $\mathbf{y} = (y_1, \ldots, y_N)$ and compute some measure for each permutation.
- For example, we might compute some measure of variable importance for each permutation.
- The idea here is that when permuting $\mathbf{y}$, the association between $\mathbf{y}$ and any "important covariates" should be lost.
- We want to see what the typical values of our variable importance measures will be when we break any association between $\mathbf{y}$ and a covariate.



- The approach of permuting the response vector can be useful in the context of difficult-to-interpret variable importance measures or variable importance measures which are known to have certain biases.
- This idea has been suggested as an alternative way of measuring variable importance for random forests (see e.g., Altmann et al. (2010) or Nembrini (2019))
- With these approaches, we permute the response vector $\mathbf{y}$ many times.

- A permutation p-value for the importance of a particular variable will be the proportion of permutations where that variable's importance score exceeded the importance score from the original data. (In this case, a smaller p-value would mean the variable was more important).

- Specifically, the permutation p-value for the importance of variable h would be given by

$$\text{p-value}_h = \frac{1}{N!}\sum_{\pi \in \mathcal{S}_N} I\Big(s_h(\mathbf{y}_{\pi}, \mathbf{X}) \geq s_h(\mathbf{y}, \mathbf{X}) \Big) \tag{5.17}$$

  where $\mathbf{y}$ denotes the vector of responses and $\mathbf{X}$ denotes the design matrix.

- Here, $s_h(\mathbf{y}, \mathbf{X})$ denotes the variable importance score for variable h when using reponse vector $\mathbf{y}$ and design matrix $\mathbf{X}$.

- Note that the formula (5.17) could be applied in the context of any method that generates a variable importance score from $\mathbf{y}$ and $\mathbf{X}$.

- Let us see an example of that if we look at a random forest model for predicting wine type from the **wine** data.

- First, we will load the data and fit a random forest model.

```
library(rattle.data)
library(randomForest)
```

```
## randomForest 4.6-14
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
wine2 <- subset(wine, Type==1 | Type==2)
wine2$Type <- factor(wine2$Type)
X <- model.matrix(Type ~ . -1, data=wine2)
yy <- wine2$Type
n <- length(yy)
nvars <- ncol(X)

## Variable importance scores using original data
originalRF <- randomForest(X, y=yy)
var.imp <- originalRF$importance
var.imp
```

```
##                  MeanDecreaseGini
## Alcohol                16.4477773
## Malic                   1.6547628
## Ash                     0.9512346
## Alcalinity              1.8280983
## Magnesium               4.2799222
## Phenols                 2.4436762
## Flavanoids              6.2584880
## Nonflavanoids           0.5184224
## Proanthocyanins         0.6201685
## Color                  10.0968269
## Hue                     0.5552606
## Dilution                1.0398252
## Proline                17.3324599
```

- Now, let us compare these original variable importance scores with the importance scores obtained across 10,000 permuted datasets.

```
nperm <- 10000
VarImpMat <- matrix(0, nrow=nperm, ncol=ncol(X))
for(k in 1:nperm) {
  ytmp <- yy[sample(1:n,size=n)]
  rf.fit <- randomForest(X, y=ytmp)
  VarImpMat[k,] <- rf.fit$importance
  ## VarImpMat[k,h] contains the importance score of
  ##  variable h in permutation k
}

perm.pval <- rep(0, nvars)
for(h in 1:nvars) {
  perm.pval[h] <- (1 + sum(VarImpMat[,h] >= var.imp[h]))/(nperm + 1)
}
```

	Permutation p-val
Alcohol	0.000
Malic	1.000
Ash	1.000
Alcalinity	1.000
Magnesium	0.923
Phenols	1.000
Flavanoids	0.080
Nonflavanoids	1.000
Proanthocyanins	1.000
Color	0.000
Hue	1.000
Dilution	1.000
Proline	0.000

Part II

Nonparametric Estimation

Chapter 6

U-Statistics

6.1 Definition

- Suppose we have i.i.d. observations $X_1, \dots, X_n$.
- U-statistics are a family of statistics used to estimate quantities that can be written as
$$\theta = E\Big\{h(X_1, \dots, X_r)\Big\} \tag{6.1}$$
- The U-statistic U which estimates (6.1) is given by the following formula:
$$U = \frac{1}{\binom{n}{r}} \sum_{c \in \mathcal{C}_{n,r}} h(X_{c_1}, \dots, X_{c_r}) \tag{6.2}$$
- The function h is usually called the **kernel** of the U-statistic. We will assume the kernel is symmetric.
- The integer r is called the **order** of the U-statistic. Typically, $r = 1$, $r = 2$, or $r = 3$ at most.
- In (6.2), $\mathcal{C}_{n,r}$ denotes the set of all $\binom{n}{r}$ combinations of size r from the set $\{1, \dots, n\}$.
- For example, if $n = 3$ and $r = 2$ then
$$\mathcal{C}_{n,r} = \{(1,2), (1,3), (2,3)\}$$
- For many common U-statistics $r = 2$ in which case (6.2) can be written as
$$U = \frac{2}{n(n-1)} \sum_{i=1}^{n} \sum_{j=i+1}^{n} h(X_i, X_j) \tag{6.3}$$

6.2 Examples

- A wide range of well-known statistics can be represented as U-statistics.

6.2.1 Example 1: The Sample Mean

- The sample mean is actually an example of a U-statistic with $r = 1$.
- Choosing $h(x) = x$ means that the corresponding U-statistic is

$$U_m = \frac{1}{n}\sum_{i=1}^{n} X_i$$

- Taking the expectation of $h(X_1)$ gives the parameter that U_m is estimating

$$E\{h(X_1)\} = E\{X_1\} = \mu$$

6.2.2 Example 2: The Sample Variance

- The sample variance is actually another example of a U-statistic. In this case, $r = 2$.
- To show why this is the case, choose the kernel $h(x_1, x_2)$ to be

$$h(x_1, x_2) = \frac{1}{2}(x_1 - x_2)^2$$

- The expectation of this kernel is $\sigma^2 = E\{h(X_1, X_2)\}$ because

$$\begin{aligned} E\{h(X_1, X_2)\} &= \frac{1}{2}\Big[E(X_1^2) - 2E(X_1)E(X_2) + E(X_2^2)\Big] \\ &= \frac{1}{2}\Big[\sigma^2 + \mu^2 - 2\mu^2 + \sigma^2 + \mu^2\Big] \\ &= \sigma^2 \end{aligned} \qquad (6.4)$$

- Also, by using formula (6.2), this choice of kernel generates the sample variance at its U-statistic:

$$
\begin{aligned}
U_{var} &= \frac{1}{\binom{n}{2}} \sum_{c \in \mathcal{C}_{n,2}} h(X_{c_1}, X_{c_2}) = \frac{2}{n(n-1)} \sum_{i=1}^{n} \sum_{j=i+1}^{n} \frac{1}{2}(X_i - X_j)^2 \\
&= \frac{2}{n(n-1)} \sum_{i=1}^{n} \sum_{j=1}^{n} \frac{1}{4}(X_i - X_j)^2 \\
&= \frac{1}{2n(n-1)} \sum_{i=1}^{n} \sum_{j=1}^{n} \{(X_i - \bar{X})^2 - 2(X_i - \bar{X})(X_j - \bar{X}) + (X_j - \bar{X})^2\} \\
&= \frac{1}{2n(n-1)} \sum_{i=1}^{n} n(X_i - \bar{X})^2 + \frac{1}{2n(n-1)} \sum_{j=1}^{n} n(X_j - \bar{X})^2 \\
&= \frac{1}{n-1} \sum_{i=1}^{n} (X_i - \bar{X})^2
\end{aligned}
$$

- Typically, the variance has the interpretation of $\sigma^2 = E\{(X_i - \mu)^2\}$. That is, σ^2 measures the expected squared deviation of X_i from its mean.

- Given the form of the U-statistic (6.4), we can also interpret the variance in the following way: if we select two observations X_i and X_j at random, the expected squared distance between X_i and X_j will be $2\sigma^2$.

- You can see this through the following computer experiment.

```r
n <- 50000
xx <- rlogis(n, location=2, scale=0.75)
diff.sq <- rep(0, 5000)
for(k in 1:5000) {
   idx <- sample(1:n, size=2)
   diff.sq[k] <- (xx[idx[1]] - xx[idx[2]])^2
}
round(var(xx), 3)
```

```
## [1] 1.854
```

```r
round(mean(diff.sq)/2, 3)
```

```
## [1] 1.874
```

6.2.3 Example 3: Gini's Mean Difference

- Gini's mean difference statistic is defined as

$$U_G = \frac{1}{\binom{n}{2}} \sum_{i=1}^{n} \sum_{j=i+1}^{n} |X_i - X_j|$$

- This is a U-statistic with $r = 2$ and kernel

$$h(X_1, X_2) = |X_1 - X_2|$$

- The parameter that we are estimating with Gini's mean difference statistic is:

$$\theta_G = E\Big\{\Big|X_1 - X_2\Big|\Big\}$$

- Gini's mean difference parameter θ_G can be interpreted in the following way: If we draw two observations at random from our population, θ_G represents the expected absolute difference between these two observations.

- The Gini coefficient θ_{Gc} is a popular measure of inequality. It is related to the mean difference parameter via

$$\theta_{Gc} = \frac{\theta_G}{2\mu},$$

where $\mu = E(X_i)$.

- **Exercise 6.1**. Compute the Gini coefficient θ_{Gc} when it is assumed that
  - $X_i \sim \text{Normal}(\mu, \sigma^2)$, for $\mu > 0$.
  - $X_i \sim \text{Exponential}(\lambda)$, (**Hint**: The difference between two independent Exponential random variables has a Laplace distribution).

6.2.4 Example 4: Wilcoxon Signed Rank Statistic

- The Wilcoxon signed rank test statistic is related to the following U statistic

$$U_{WS} = \frac{2}{n(n-1)} \sum_{i=1}^{n} \sum_{j=i+1}^{n} I\Big(X_i + X_j \geq 0\Big)$$

- U_{WS} is a U-statistic of order 2 with kernel

$$h(x, y) = I\Big(x + y \geq 0\Big)$$

- Hence, U_{WS} can be interpreted as an estimate of the following parameter

$$\theta_{WS} = P\Big(X_i + X_j \geq 0\Big) = P\Big(X_i \geq -X_j\Big)$$

- If the distribution of X_i is symmetric around 0, θ_{WS} will be equal to 1/2.

- Recall that the Wilcoxon signed rank test is designed to detect distributions which are not symmetric around 0.
- The Wilcoxon signed rank statistic T_n that we defined in Chapter 3 had the following formula

$$T_n = \sum_{i=1}^{n} \text{sign}(X_i) R_i(|\mathbf{X}|)$$

- Additional algebra can show that

$$\begin{aligned} T_n &= n(n-1)U_{WS} + 2\sum_{i=1}^{n} I(X_i > 0) - \frac{n(n+1)}{2} \\ &= n(n-1)U_{WS} + 2S_n - \frac{n(n+1)}{2} \end{aligned} \tag{6.5}$$

where S_n is the sign test statistic defined in Chapter 3.

- For large n, T_n is largely determined by $U_{WS} - 1/2$. Hence, a "large" value of $U_{WS} - 1/2$ will lead to rejection of the one-sample null hypothesis discussed in Section 3.3.

- If you want to derive (6.5) (though you don't need to know how), I think it is helpful to note the following

$$\begin{aligned} I(X_{(i)} > 0)R_{(i)}(|\mathbf{X}|) &= \sum_{j=1}^{n} I(X_{(i)} > 0) I(|X_{(i)}| \geq |X_j|) = \sum_{j=1}^{n} I(X_{(i)} \geq |X_j|) \\ &= \sum_{j=1}^{n} I(X_{(i)} \geq |X_{(j)}|) = \sum_{j=1}^{i} I(X_{(i)} \geq -X_{(j)}) \\ &= \sum_{j=1}^{i} I(X_{(i)} + X_{(j)} \geq 0) \end{aligned}$$

6.3 Inference using U-statistics

- By using a large-sample approximation, you can construct a confidence interval for your U-statistic parameter of interest θ where

$$\theta = E\Big\{h(X_1, \dots, X_r)\Big\} \tag{6.6}$$

- While the U-statistic is a sum of random variables that are not necessarily independent, there is a type of Central Limit Theorem for U-statistics.
- Specifically, under appropriate regularity conditions:

$$\sqrt{n}(U - \theta) \longrightarrow \text{Normal}\Big(0, r^2\varphi\Big)$$

- The formula for φ is

$$\varphi = \text{Cov}\Big(h(X_1, X_2, \dots, X_r), h(X_1, X_2', \dots, X_r')\Big),$$

  where $X_1', X_2', \dots, X_r'$ are thought of as another i.i.d. sample from the same distribution as $X_1, \dots, X_r$.

6.4 U-statistics for Two-Sample Problems

- In two-sample problems, we have data from two groups which we label $X_1, \dots, X_n$ and $Y_1, \dots, Y_m$
- A U-statistic with order (r, s) for a two-sample problem is

$$U = \frac{1}{\binom{n}{r}} \frac{1}{\binom{m}{s}} \sum_{c \in \mathcal{C}_{n,r}} \sum_{q \in \mathcal{C}_{m,s}} h(X_{c_1}, \dots, X_{c_r}, Y_{q_1}, \dots, Y_{q_s}) \tag{6.7}$$

6.4.1 The Mann-Whitney Statistic

- Consider the following U-statistic

$$U_{MW} = \frac{1}{mn} \sum_{i=1}^{n} \sum_{j=1}^{m} I(X_i \geq Y_j) \tag{6.8}$$

- This is a U-statistic of order $(1, 1)$ with kernel $h(x, y) = I(x \geq y)$.
- Hence, the U-statistic U_{MW} can be thought of as an estimate of the following parameter

$$\theta_{MW} = P\Big(X_i \geq Y_j\Big) \tag{6.9}$$

- If both X_i and Y_j have the same distribution, then θ_{MW} should equal 1/2.

- The statistic mnU_{MW} is known as the **Mann-Whitney** statistic.
- The Mann-Whitney statistic has a close relation to the Wilcoxon rank sum statistic W that we defined in Section 3.2:

$$\begin{aligned} mnU_{MW} &= \sum_{i=1}^{n}\sum_{j=1}^{m} I(X_i \geq Y_j) \\ &= \sum_{i=1}^{n}\Big[\sum_{j=1}^{m} I(X_i \geq Y_j) + \sum_{k=1}^{n} I(X_i \geq X_k)\Big] - \sum_{i=1}^{n}\sum_{k=1}^{n} I(X_i \geq X_k) \\ &= \sum_{i=1}^{n} R_i(\mathbf{Z}) - \sum_{i=1}^{n} R_i(\mathbf{X}) \qquad (6.10) \\ &= W - \frac{n(n+1)}{2} \end{aligned}$$

- In other words, the Mann-Whitney statistic is equal to the WRS statistic minus a constant term.
- In (6.10), we are defining $\mathbf{Z}$ as the pooled-data vector $\mathbf{Z} = (X_1, \dots, X_n, Y_1, \dots, Y_m)$.
- Also, the above derivation assumes no ties so that $\sum_{i=1}^{n} R_i(\mathbf{X}) = n(n+1)/2$.

- Because $W = n(n+1)/2 + mnU_{MW}$ is a linear function of U_{MW}, inference from the Wilcoxon rank sum test (when using large-sample p-values) should match inferences made from using U_{MW} to test the hypothesis $H_0 : \theta_{MW} = 1/2$.
- In other words, the two-sided Wilcoxon rank sum test can be thought of as a test of $H_0 : \theta_{MW} = 1/2$ vs. $H_A : \theta_{MW} \neq 1/2$, where θ_{MW} is the parameter defined in (6.9).

6.5 Measures of Association

- Many important measures of association are also examples of U-statistics.

- For measures of association, we have observations on n pairs of variables

$$(X_1, Y_1), \dots, (X_n, Y_n),$$

  and our goal is to report some measure which quantifies the relationship between these two variables.

- In this context, we will think about U-statistics which have the form

$$U = \frac{1}{\binom{n}{r}} \sum_{c \in \mathcal{C}_{n,r}} h\left(\begin{bmatrix} X_{c_1} \\ Y_{c_1} \end{bmatrix}, \dots, \begin{bmatrix} X_{c_r} \\ Y_{c_r} \end{bmatrix} \right) \tag{6.11}$$

6.5.1 Spearman's Rank Correlation

- Spearman's sample rank correlation is defined as

$$\begin{aligned} \hat{\rho}_R &= \frac{\sum_{i=1}^n \{R_i(\mathbf{X}) - \bar{R}(\mathbf{X})\}\{R_i(\mathbf{Y}) - \bar{R}(\mathbf{Y})\}}{\left[\sum_{i=1}^n \{R_i(\mathbf{X}) - \bar{R}(\mathbf{X})\}^2 \sum_{i=1}^n \{R_i(\mathbf{Y}) - \bar{R}(\mathbf{Y})\}^2\right]^{1/2}} \\ &= \frac{12}{n(n-1)(n+1)} \sum_{i=1}^n R_i(\mathbf{X})R_i(\mathbf{Y}) - \frac{3(n+1)}{n-1}, \end{aligned} \tag{6.12}$$

  where $\bar{R}(X) = \frac{1}{n}\sum_{i=1}^n R_i(\mathbf{X})$ and $\bar{R}(\mathbf{Y}) = \frac{1}{n}\sum_{i=1}^n R_i(\mathbf{Y})$.

- Remember that $R_i(\mathbf{X})$ denotes the rank of X_i when only using the vector $\mathbf{X} = (X_1, \dots, X_n)$ to compute the rankings. Likewise, $R_i(\mathbf{Y})$ denotes the rank of Y_i when only using the vector $\mathbf{Y} = (Y_1, \dots, Y_n)$ to compute the rankings.

- Notice that $\hat{\rho}_R$ comes from applying the usual Pearson's estimate of correlation to the ranks $(R_i(\mathbf{X}), R_i(\mathbf{Y}))$ rather than the original data (X_i, Y_i).

- As with the usual estimate of correlation, $\hat{\rho}_R$ is large (i.e., closer to 1) whenever large values of X_i tend to be associated with large values of Y_i. Similarly, $\hat{\rho}_R$ is small wheneve large values of X_i tend to be associated with small values of Y_i.

- Values of $\hat{\rho}_R$ near zero indicate that there is little association between these two variables.

- Due to its use of ranks, $\hat{\rho}_R$ is less sensitive to outliers than Pearson's correlation.

- Another important feature of $\hat{\rho}_R$ is that it is invariant to monotone transformations of the data.

- While Pearson's correlation is very effective for detecting linear associations between two variables, the rank correlation is very effective at detecting any monotone associations between two variables.
- $\hat{\rho}_R$ will equal 1 if Y_i is a monotone increasing function of X_i, and $\hat{\rho}_R$ will equal -1 if Y_i is a monotone decreasing function X_i.

```
xx <- pmax(rnorm(100, mean=10), 0.01)
yy <- pmax(xx + rnorm(100, sd=.5), 0.01)

## Compare the usual Pearson's correlation between
## (xx, yy) and (xx, yy^2)
round( c( cor(xx, yy), cor(xx, yy^2)), 3)
```

```
## [1] 0.893 0.890
```

```
## Now do the same for Spearman's rank correlation
round(c( cor(xx, yy, method="spearman"),
         cor(xx, yy^2, method="spearman")), 3)
```

```
## [1] 0.867 0.867
```

- $\hat{\rho}_R$ can be thought of as an estimate of the following population quantity:

$$\theta_R = 12P\Big(X_1 \geq X_2, Y_1 \geq Y_3\Big) - 3 \tag{6.13}$$

- To justify this, first notice that

$$\begin{aligned} V_R &= \frac{1}{n^3}\sum_{i=1}^n R_i(\mathbf{X})R_i(\mathbf{Y}) = \frac{1}{n^3}\sum_{i=1}^n\sum_{j=1}^n I(X_i \geq X_j)\sum_{k=1}^n I(Y_i \geq Y_k) \\ &= \frac{1}{n^3}\sum_{i=1}^n\sum_{j=1}^n\sum_{k=1}^n I(X_i \geq X_j)I(Y_i \geq Y_k) \end{aligned}$$

- While V_R is not exactly a U-statistic, it can be thought of as roughly a "U-statistic" with non-symmetric kernel function

$$h\left(\begin{bmatrix} X_1 \\ Y_1 \end{bmatrix}, \begin{bmatrix} X_1 \\ Y_1 \end{bmatrix}, \begin{bmatrix} X_3 \\ Y_3 \end{bmatrix}\right) = I(X_1 \geq X_2)I(Y_1 \geq Y_3)$$

- So, we should expect V_R to converge to $P\{X_1 \geq X_2, Y_1 \geq Y_3\}$ as n gets larger.

- Using (6.12) and our formula for V_R, we can write $\hat{\rho}_R$ as

$$\begin{aligned}\hat{\rho}_R &= \frac{12}{n(n-1)(n+1)}\sum_{i=1}^{n} R_i(\mathbf{X})R_i(\mathbf{Y}) - \frac{3(n+1)}{n-1} \\ &= 12V_R\Big(\frac{n^3}{n(n-1)(n+1)}\Big) - \frac{3(n+1)}{n-1}.\end{aligned}$$

Exercise 6.2. Why does θ_R equal zero when X_i and Y_i are independent? Why is $-1 \leq \theta_R \leq 1$?

6.5.2 Kendall's tau

- Ignoring the possibility of ties, Kendall's τ-statistic U_τ is given by

$$U_\tau = \frac{2}{n(n-1)}\sum_{i=1}^{n}\sum_{j=i+1}^{n}\Big[2\times I\Big\{(X_j - X_i)(Y_j - Y_i) > 0\Big\} - 1\Big]$$

- Note that U_τ is a U-statistic of order 2 with kernel

$$h\left(\begin{bmatrix} X_1 \\ Y_1\end{bmatrix}, \begin{bmatrix} X_2 \\ Y_2\end{bmatrix}\right) = 2\times I\Big\{(X_2 - X_1)(Y_2 - Y_1) > 0\Big\} - 1 \qquad (6.14)$$

- Assuming the probability of ties is zero, Kendall's τ can be thought of as an estimate of the following quantity

$$\theta_\tau = 2P\Big\{(X_j - X_i)(Y_j - Y_i) > 0\Big\} - 1 \qquad (6.15)$$

- Kendall's τ must be in between -1 and 1.
- If X_i and Y_i are idependent, Kendall's τ will be equal to zero (why?).

- In the context of computing U_τ, pairs of observations (X_i, Y_i) and (X_j, Y_j) are said to be **concordant** if the sign of $X_j - X_i$ agrees with the sign of $Y_j - Y_i$.
- If the sign of $X_j - X_i$ and $Y_j - Y_i$ do **not** agree, then the pairs (X_i, Y_i) and (X_j, Y_j) are said to be **discordant**.

- If either $X_j = X_i$ or $Y_j = Y_i$, then the pairs (X_i, Y_i) and (X_j, Y_j) are neither concordant or discordant.

- Let us define the following

$$\begin{aligned} n_c &= \text{the number of concordant pairs} \\ n_d &= \text{the number of discordant pairs} \\ n_n &= \text{number of pairs which are neither} \end{aligned}$$

- Here, we are counting n_c and n_d from the number of unique possible pairings. There are $n(n-1)/2$ unique pairings, and hence

$$n_c + n_d + n_n = \binom{n}{2} = \frac{n(n-1)}{2} \tag{6.16}$$

- Notice that n_c can be expressed in terms of indicator functions as

$$n_c = \sum_{i=1}^{n} \sum_{j=i+1}^{n} I\Big\{(X_j - X_i)(Y_j - Y_i) > 0\Big\}$$

- If we assume that there are no ties (i.e., $n_n = 0$), then U_τ can be written as

$$U_\tau = \frac{4n_c}{n(n-1)} - 1 = \frac{2n_c + 2n_c - n(n-1)}{n(n-1)} = \frac{2n_c - 2n_d}{n(n-1)} = \frac{2(n_c - n_d)}{n(n-1)}$$

- Under independence, the number of concordant and discordant pairs should be roughly equal.

- We just need ordinal data to use Kendall's τ. Kendall's τ can be computed as long you can tell if one observation is "larger" than another.

- Kendall's τ is often used in the context of assessing the agreement between different ratings.

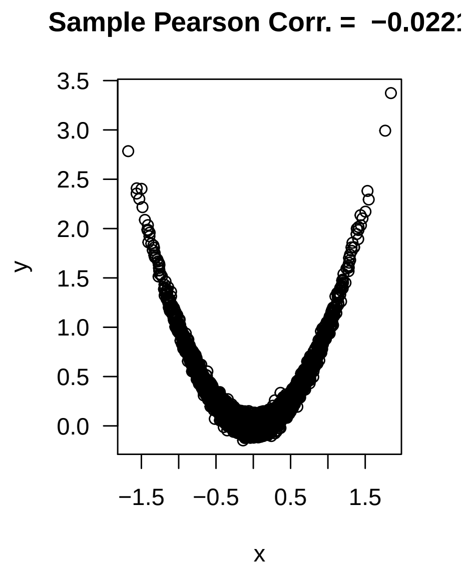
- In this context, we might have K different judges which are rating J different objects. If r_{jk} denotes the object-j rating given by judge k, Kendall's τ from the J pairs $(r_{11}, r_{12}), \ldots, (r_{J1}, r_{J2})$ would give a measure of the agreement between judges 1 and 2.

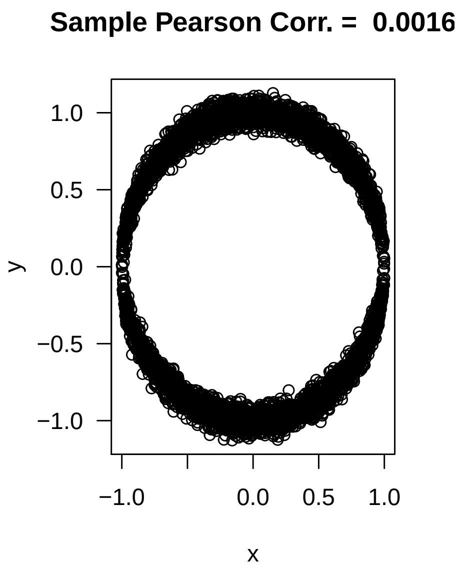
6.5.3 Distance Covariance and Correlation

- A correlation equal to zero does not imply that two random variables are independent.
- For example, if $X \sim \text{Normal}(0, 1)$, then

$$\text{Corr}(X, X^2) = \text{Cov}(X, X^2) = E(X^3) = 0$$

- This is also true for Spearman's rank correlation and Kendall's τ. You can have situations where $\theta_R = 0$ but X and Y are not independent. Similarly, you can have situations where $\theta_\tau = 0$ while X and Y are not independent.
- Note that the association between the two variables in the figures below is **non-monotone**.





6.5.3.1 Definition

- **Distance covariance** and **distance correlation** are two measures of dependence that have been developed much more recently (see Székely et al. (2007)).
- The interesting thing about these two measures is that: if they equal zero then it implies that the two random variables are independent.

- Moreover, the measures have a relatively straightforward formula, and they have easily computable estimates.
- For i.i.d. bivariate random variables $(X_1, Y_1), \ldots, (X_n, Y_n)$, the squared distance covariance parameter is defined as

$$\begin{aligned}\theta^2_{dCov,XY} &= E\Big\{|X_1 - X_2||Y_1 - Y_2|\Big\} + E\Big\{|X_1 - X_2|\Big\}E\Big\{|Y_1 - Y_2|\Big\} \\ &- 2E\Big\{|X_1 - X_2||Y_1 - Y_3|\Big\}\end{aligned}$$

- The distance correlation bettween X_i and Y_i is then defined as

$$\rho_{d,XY} = \frac{\theta_{dCov,XY}}{\theta_{dCov,XX}\theta_{dCov,YY}}$$

- Notice that we must have $\theta_{dCov,XY} \geq 0$ and $\rho_{d,XY} \geq 0$.
- There is no notion of a negative correlation when using distance correlation.
- The interpretation of $\theta^2_{dCov,XY}$ is perhaps not as clear as the usual correlation parameter. Nevertheless, $\theta_{dCov,XY} = 0$ implies independence, and larger values of $\theta_{dCov,XY}$ imply that X_i and Y_i have some form of "greater" association.

Example

- Let us consider the example we had before where we compared X and X^2.
- Specifically, suppose we have observed pairs $(X_1, Y_1), \ldots, (X_n, Y_n)$ where $X_i \sim \text{Normal}(0, 1)$ and $Y_i = X_i^2$.
- In this case, the distance covariance turns out to be

$$\begin{aligned}&\theta^2_{dCov,XY} = E\Big\{|X_1 - X_2||Y_1 - Y_2|\Big\} + E\Big\{|X_1 - X_2|\Big\}E\Big\{|Y_1 - Y_2|\Big\} - 2E\Big\{|X_1 - X_2||Y_1 - Y_3|\Big\} \\ &= E\Big\{|X_1 - X_2||X_1^2 - X_2^2|\Big\} + E\Big\{|X_1 - X_2|\Big\}E\Big\{|X_1^2 - X_2^2|\Big\} - 2E\Big\{|X_1 - X_2||X_1^2 - X_3^2|\Big\}\end{aligned}$$

- It could be a lot work to compute the above expectation exactly. However, we can estimate it pretty closely using simulation:

```r
set.seed(4157)
nreps <- 500000 ## number of simulation replications
term1 <- term2 <- term3 <- term4 <- rep(0, nreps)
for(k in 1:nreps) {
    xx <- rnorm(3)
    term1[k] <- abs(xx[1] - xx[2])*abs(xx[1]^2 - xx[2]^2)
    term2[k] <- abs(xx[1] - xx[2])
    term3[k] <- abs(xx[1]^2 - xx[2]^2)
    term4[k] <- abs(xx[1] - xx[2])*abs(xx[1]^2 - xx[3]^2)
}
dcov.sq.est <- mean(term1) + mean(term2)*mean(term3) - 2*mean(term4)
dcov.sq.est
```

```
## [1] 0.137895
```

- The squared distance covariance for this example seems to be about 0.14.
- Thus, the distance covariance is positive for this example where the two variables are dependent while the usual covariance between these two variables is zero.

- **Exercise 6.2**. For the example where we have observed pairs $(X_1, Y_1), \ldots, (X_n, Y_n)$ with $X_i \sim \text{Normal}(0, 1)$ and $Y_i = X_i^2$, compute Kendall's τ parameter θ_τ.

6.5.3.2 Estimation of Distance Covariance and Distance Correlation

- The distance covariance and correlation are estimated by using a bunch of pairwise distances between our observations.
- The pairwise distances a_{ij} and b_{ij} for the X_i and Y_i are defined as

$$\begin{aligned} a_{ij} &= |X_i - X_j| \\ b_{ij} &= |Y_i - Y_j| \end{aligned}$$

- We then construct the $n \times n$ matrix $\mathbf{A}$ (with elements A_{ij}) and the $n \times n$ matrix $\mathbf{B}$ (with elements B_{ij}) in the following way

$$A_{ij} = \begin{cases} a_{ij} - \frac{1}{n-2}a_{i.} - \frac{1}{n-2}a_{.j} + \frac{1}{(n-1)(n-2)}a_{..} & \text{if } i \neq j \\ 0 & \text{if } i = j \end{cases}$$

$$B_{ij} = \begin{cases} b_{ij} - \frac{1}{n-2}b_{i.} - \frac{1}{n-2}b_{.j} + \frac{1}{(n-1)(n-2)}b_{..} & \text{if } i \neq j \\ 0 & \text{if } i = j \end{cases}$$

where $a_{i.} = \sum_{k=1}^{n} a_{ik}$, $a_{.j} = \sum_{k=1}^{n} a_{kj}$, and $a_{..} = \sum_{k=1}^{n}\sum_{l=1}^{n} a_{ij}$.

- In other words, **A** is a matrix containing "centered" pairwise distances.

- The estimate of the squared distance covariance parameter is then given by

$$\hat{\theta}_{dCov,XY}^2 = \frac{1}{n(n-3)} \sum_{i=1}^{n}\sum_{j=1}^{n} A_{ij}B_{ij} \tag{6.17}$$

- The estimate of the distance correlation is

$$\hat{\rho}_{d,XY} = \frac{\hat{\theta}_{dCov,XY}}{\hat{\theta}_{dCov,XX}\hat{\theta}_{dCov,YY}} \tag{6.18}$$

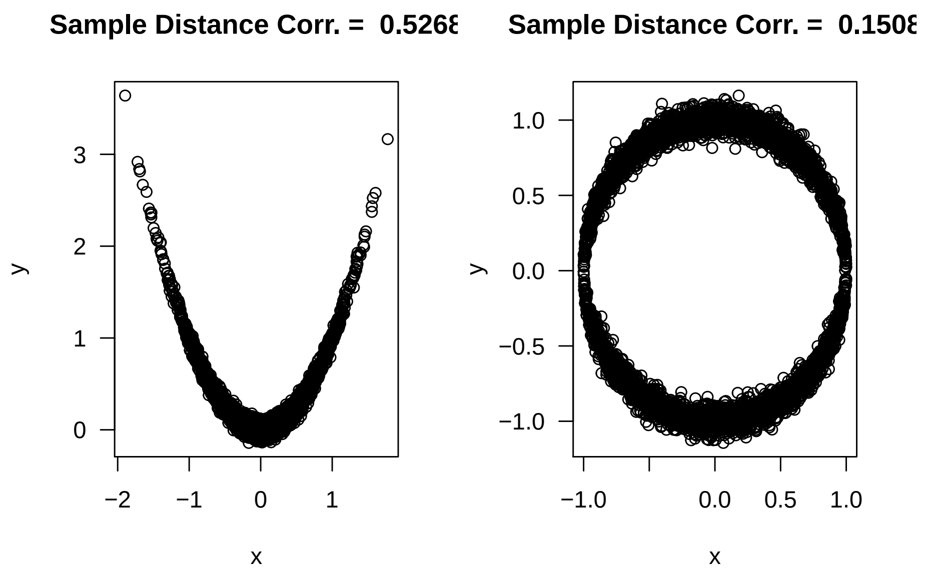
- It turns out that $\hat{\theta}_{dCov,XY}^2$ is a U-statistic of order 4 (see Huo and Székely (2016) for a justification of this). It has kernel function

$$h\left(\begin{bmatrix} X_1 \\ Y_1 \end{bmatrix}, \begin{bmatrix} X_1 \\ Y_1 \end{bmatrix}, \begin{bmatrix} X_3 \\ Y_3 \end{bmatrix}, \begin{bmatrix} X_4 \\ Y_4 \end{bmatrix} \right) = \frac{1}{4}\sum_{i=1}^{4}\sum_{j=1}^{4} A_{ij}B_{ij} \tag{6.19}$$

- You can compute distance covariances and distance correlations using the **energy** package in **R**.

```r
library(energy)

n <- 5000
## generate "parabola" data
xx1 <- rnorm(n, sd=0.5)
yy1 <- xx1^2 + rnorm(n, sd=0.05)

## generate circle data
xx2 <- runif(n, min=-1, max=1)
yy2 <- sample(c(-1,1), size=n, replace=TRUE)*sqrt(1 - xx2^2) + rnorm(n, sd=.05)

d.cor1 <- dcor(xx1, yy1)
d.cor2 <- dcor(xx2, yy2)
```

```
par(mfrow=c(1,2))
plot(xx1, yy1, xlab="x", ylab="y", main=paste("Sample Distance Corr. = ",
                                              round(d.cor1 ,4)), las=1)
plot(xx2, yy2, xlab="x", ylab="y", main=paste("Sample Distance Corr. = ",
                                              round(d.cor2, 4)), las=1)
```



```
## Let's just compare the values of distance correlation and Pearson's
## for both examples
p.cor1 <- cor(xx1, yy1)
p.cor2 <- cor(xx2, yy2)

kend.cor1 <- cor(xx1, yy1, method="kendall")
kend.cor2 <- cor(xx2, yy2, method="kendall")

spear.cor1 <- cor(xx1, yy1, method="spearman")
spear.cor2 <- cor(xx2, yy2, method="spearman")

# Pearson, Kendall's-tau, Rank, Distance Correlation
round(c(p.cor1, kend.cor1, spear.cor1, d.cor1), 4) ## parabola
```

```
## [1] -0.0124  0.0077  0.0109  0.5268
```

```r
round(c(p.cor2, kend.cor2, spear.cor2, d.cor2), 4) ## circle
```

```
## [1] 0.0134 0.0041 0.0138 0.1508
```

Chapter 7

The Empirical Distribution Function

7.1 Definition and Basic Properties

- Every random variable has a cumulative distribution function (cdf).
- The cdf of a random variable X is defined as

$$F(t) = P(X \leq t) \tag{7.1}$$

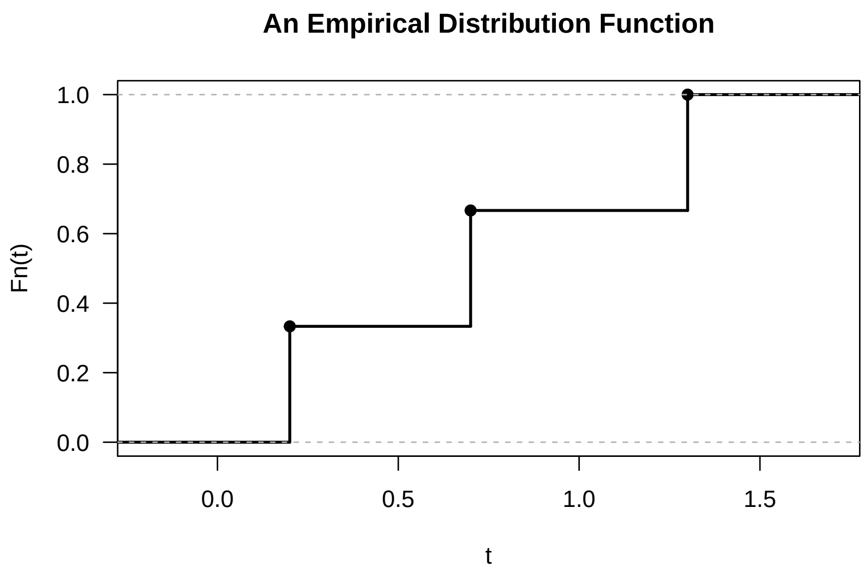
- The empirical distribution function or empirical cumulative distribution function (ecdf) estimates $F(t)$ by computing the proportion of observations which are less than or equal to t.
- For i.i.d. random variables $X_1, \ldots, X_n$ with cdf F, the empirical distribution function is defined as

$$\hat{F}_n(t) = \frac{1}{n}\sum_{i=1}^{n} I(X_i \leq t)$$

- Note that the empirical distribution function can be computed for any type of data without making any assumptions about the distribution from which the data arose.
- The only assumption we are making is that $X_1, \ldots, X_n$ constitute an i.i.d. sample from some common distribution function F.

- For example, if we observed $X_1 = 0.7$, $X_2 = 0.2$, and $X_3 = 1.3$, the corresponding empirical distribution function would be

$$\hat{F}_3(t) = \begin{cases} 0 & \text{for } t < 0.2 \\ 1/3 & \text{for } 0.2 \leq t < 0.7 \\ 2/3 & \text{for } 0.7 \leq t < 1.3 \\ 1 & \text{for } t \geq 1.3 \end{cases} \tag{7.2}$$



7.2 Confidence intervals for F(t)

- For a fixed value of t, the distribution of $n\hat{F}_n(t)$ is

$$n\hat{F}_n(t) \sim \text{Binomial}(n, F(t)) \tag{7.3}$$

- This is because, for a fixed t, $n\hat{F}_n(t)$ is the sum of n independent Bernoulli random variables $W_1^t, \dots, W_n^t$

$$n\hat{F}_n(t) = \sum_{i=1}^{n} W_i^t = \sum_{i=1}^{n} I(X_i \leq t) \tag{7.4}$$

- The probability that $W_i^t = 1$ is

$$P(W_i^t = 1) = P(X_i \leq t) = F(t)$$

Pointwise Confidence Intervals

- Because $\hat{F}_n(t)$ is a mean of independent random variables, we can say that

$$\frac{\sqrt{n}\Big(\hat{F}_n(t) - F(t)\Big)}{\sqrt{\hat{F}_n(t)(1 - \hat{F}_n(t))}} \longrightarrow \text{Normal}\Big(0, 1\Big)$$

- The above asymptotic statement is the basis for constructing **pointwise confidence intervals** for $F(t)$.
- For a fixed t, a $100 \times (1 - \alpha)\%$ confidence interval for $F(t)$ is the following

$$\begin{aligned} CI_{\alpha}^{pw}(t) &= [L_{\alpha}^{pw}(t), U_{\alpha}^{pw}(t)] \\ L_{\alpha}^{pw}(t) &= \max\left\{ \hat{F}_n(t) - z_{1-\alpha/2}\sqrt{\frac{\hat{F}_n(t)(1 - \hat{F}_n(t))}{n}}, 0 \right\} \\ U_{\alpha}^{pw}(t) &= \min\left\{ \hat{F}_n(t) + z_{1-\alpha/2}\sqrt{\frac{\hat{F}_n(t)(1 - \hat{F}_n(t))}{n}}, 1 \right\} \end{aligned} \qquad (7.5)$$

- Plotting $CI_{\alpha}^{pw}(t)$ for different values of t, would give pointwise confidence intervals for the distribution function. Plotting pointwise confidence intervals for $F(t)$ or for survival functions $S(t) = 1 - F(t)$ is fairly common in practice.
- However, these pointwise confidence intervals only hold for each point separately.

Simultaneous Confidence Bands

- Simultaneous confidence bands can be thought of as two functions $L_{\alpha}^{band}(t)$ and $U_{\alpha}^{band}(t)$ such that we are "$100 \times (1 - \alpha)\%$ confident" that all of $F(t)$ is contained within the bands $L_{\alpha}^{band}(t)$ and $U_{\alpha}^{band}(t)$.
- Specifically, we want the statement

$$L_{\alpha}^{band}(t) \leq F(t) \leq U_{\alpha}^{band}(t) \quad \text{for all } t \qquad (7.6)$$

  to hold with at least $1 - \alpha$ probability.
- In other words, we want less than α probability for any part of the path of $F(t)$ going outside of the bands.

- One choice of $L_{\alpha}^{band}(t)$ and $U_{\alpha}^{band}(t)$ which has this property is the following

$$L_{\alpha}^{band}(t) = \max\{\hat{F}_n(t) - \delta_{\alpha,n}, 0\} \qquad U_{\alpha}^{band}(t) = \min\{\hat{F}_n(t) + \delta_{\alpha,n}, 1\}, \tag{7.7}$$

where $\delta_{\alpha,n}$ is given by

$$\delta_{\alpha,n} = \sqrt{\frac{1}{2n}\ln\Big(\frac{2}{\alpha}\Big)}$$

- The reason this choice of confidence band works is the Dvoretzky-Kiefer-Wolfowitz (DKW) inequality. The DKW inequality states that

$$P\left(\sup_t |F(t) - \hat{F}_n(t)| > \varepsilon\right) \leq 2e^{-2n\varepsilon^2}$$

- Our choice of confidence bands (7.7) then works because

$$\sup_t |F(t) - \hat{F}_n(t)| \leq \delta_{\alpha,n}$$

is equivalent to

$$L_{\alpha}^{band}(t) \leq F(t) \leq U_{\alpha}^{band}(t) \qquad \text{for all } t$$

- Then, from the DKW inequality we have

$$\begin{aligned} P\left(L_{\alpha}^{band}(t) \leq F(t) \leq U_{\alpha}^{band}(t) \quad \text{for all } t\right) &= P\left(\sup_t |F(t) - \hat{F}_n(t)| \leq \delta_{n,\alpha}\right) \\ &\geq 1 - 2e^{-2n\delta_{\alpha,n}^2} \\ &= 1 - \alpha. \end{aligned}$$

- Confidence bands will almost always be wider than the pointwise confidence intervals.

- This extra width is due to the fact that we are requiring the coverage probability to hold for the entire path of $F(t)$ rather than at just a single point.

7.3 The Empirical Distribution Function in R

- We will see how to work with empirical distribution functions in **R** by using data from a study on kidney function.

- This dataset has 157 observations which has the age of each study participant and a measure of overall kidney function. The data can be obtained at https://web.stanford.edu/~hastie/CASI_files/DATA/kidney.txt

- We will only look at the **tot** variable in this chapter.

```
kidney <- read.table("https://web.stanford.edu/~hastie/CASI_files/DATA/kidney.txt",
                     header=TRUE)
head(kidney)
```

```
##   age   tot
## 1  18  2.44
## 2  19  3.86
## 3  19 -1.22
## 4  20  2.30
## 5  21  0.98
## 6  21 -0.50
```

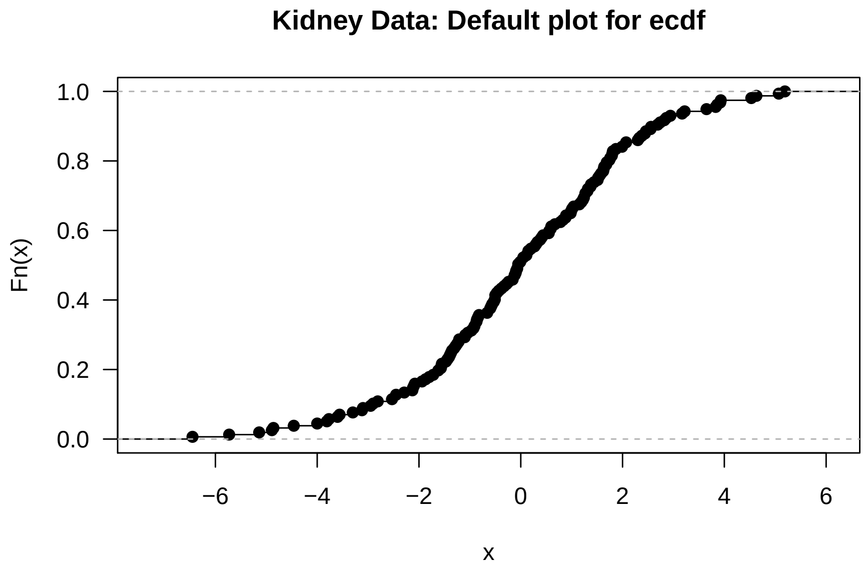
- The **ecdf** function is the main function which computes the empirical distribution function in **R**

- The **ecdf** function will create an **ecdf** object. To create an ecdf object for the kidney totals, use the following code:

```
kidney.Fhat <- ecdf(kidney$tot)
```

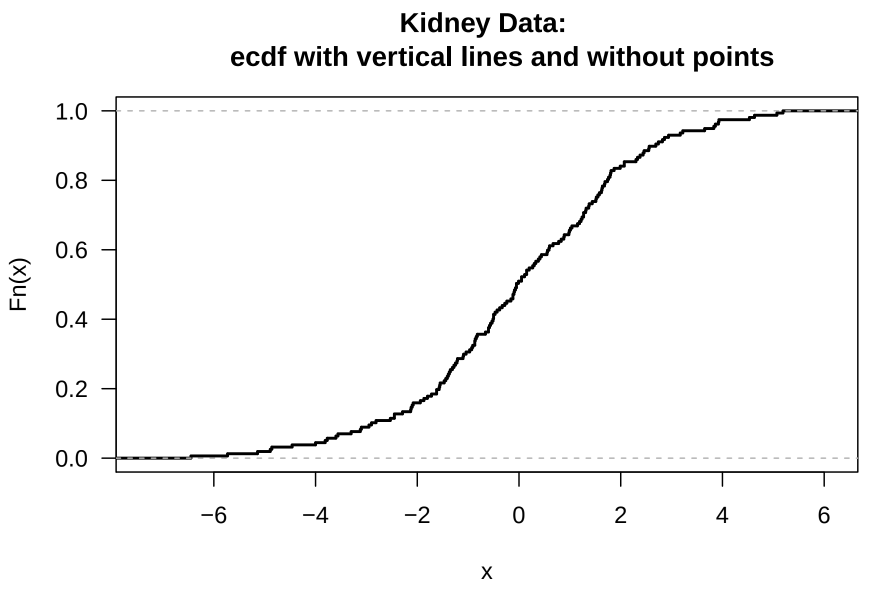
- You can plot the ecdf for the kidney totals by just calling **plot(ecdf)**

```
plot(kidney.Fhat, main = "Kidney Data: Default plot for ecdf", las=1)
```



- If you don't like the look of the points in the ecdf plot, you can use add the argument **do.points** = **FALSE** when calling plot. Also, you can add the argument **verticals** =**TRUE** if you want the plot to draw vertical lines whenever there is a jump in the empirical distribution function.

```
plot(kidney.Fhat, do.points=FALSE, verticals=TRUE, main = "Kidney Data:
    ecdf with vertical lines and without points", las=1, lwd=2)
```



- A nice feature of of the **ecdf** function is that **ecdf** object can be treated as a function which computes the empirical distribution function. For example,

```
kidney.Fhat <- ecdf(kidney$tot)

kidney.Fhat(0)
```

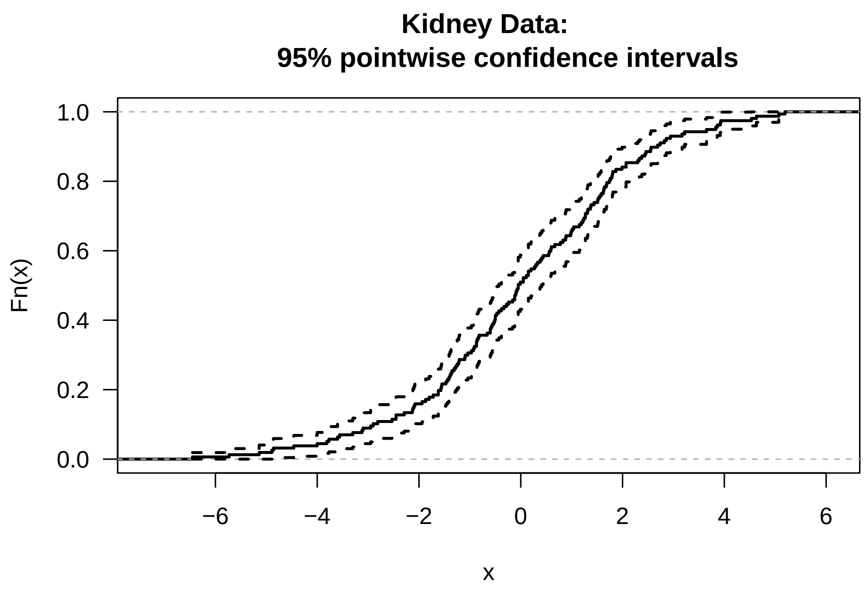
```
## [1] 0.5095541
```

```
kidney.Fhat( c(-1,1,4) )
```

```
## [1] 0.3057325 0.6560510 0.9745223
```

- **R** does not plot confidence intervals when plotting the empirical distribution function.
- We can do this ourselves, by using the pointwise confidence interval formula shown in (7.5)

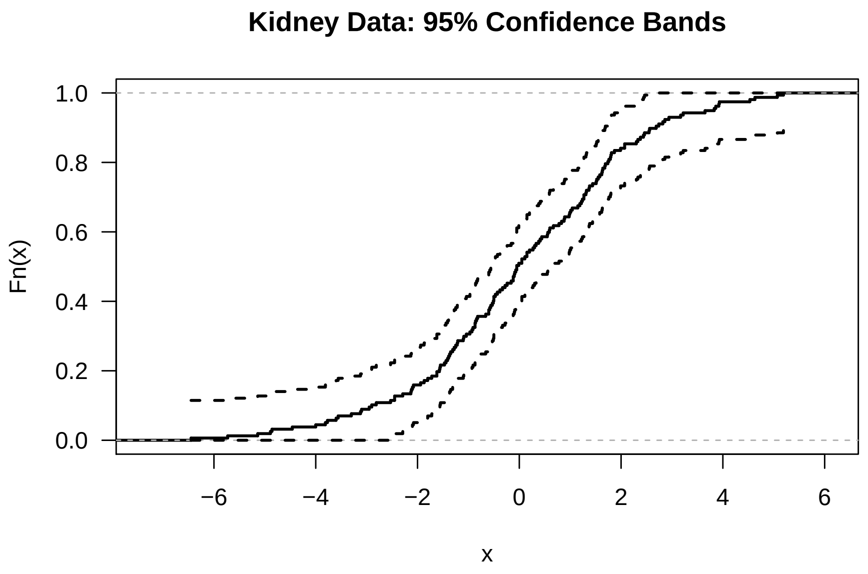
```
## 1. First, we will compute the standard errors at each of the
##    observed time points
tt <- sort(unique(kidney$tot))
std.err <- sqrt(kidney.Fhat(tt)*(1 - kidney.Fhat(tt))/ length(kidney$tot))

## 2. Now, compute the confidence intervals at each time point
ci.low <- pmax(kidney.Fhat(tt) - qnorm(.975)*std.err, 0)
ci.upper <- pmin(kidney.Fhat(tt) + qnorm(.975)*std.err, 1)

## 3. Now, plot the results. Note that type="s" in the lines function produces
##    "step functions" which pass through the provided points.
plot(kidney.Fhat, do.points=FALSE, verticals=TRUE, main = "Kidney Data:
     95% pointwise confidence intervals", las=1, lwd=2)
lines(tt, ci.low, type="s", lty=2, lwd=2)
lines(tt, ci.upper, type="s", lty=2, lwd=2)
```



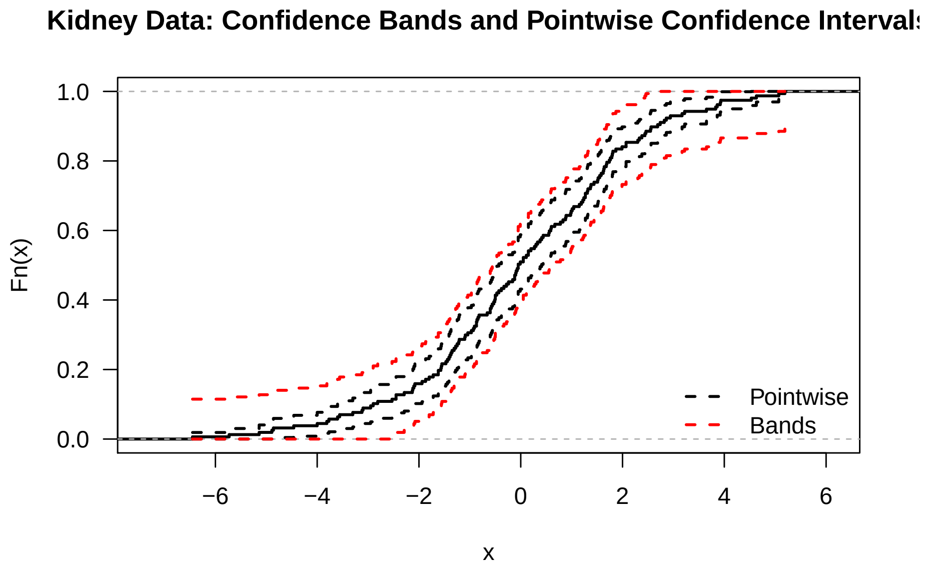
- We could plot the confidence bands as well.

```
n <- length(kidney$tot)

## Compute the confidence bands at each time point
ci.band.low <- pmax(kidney.Fhat(tt) - sqrt(log(2/0.05)/(2*n)), 0)
ci.band.upper <- pmin(kidney.Fhat(tt) + sqrt(log(2/0.05)/(2*n)), 1)
```

```
plot(kidney.Fhat, do.points=FALSE, verticals=TRUE,
    main = "Kidney Data: 95% Confidence Bands", las=1, lwd=2)
lines(tt, ci.band.low, type="s", lty=2, lwd=2)
lines(tt, ci.band.upper, type="s", lty=2, lwd=2)
```



- Comparing the pointwise confidence intervals and the simultaneous confidence bands in the same plot shows how much wider the confidence bands are:



7.4 The Kolmogorov-Smirnov Test

- The one-sample Kolmogorov-Smirnov (KS) test is a type of goodness-of-fit test that is based on the empirical distribution function.
- The KS test will test whether or not our data $X_1, \ldots, X_n$ comes from a specific distribution of interest F_0.
- Supposing our data $X_1, \ldots, X_n$ are an i.i.d. sample with distribution function F, the hypothesis test of interest can be stated as

$$H_0 : F = F_0 \qquad \text{vs.} \qquad H_A : F \neq F_0 \tag{7.8}$$

- The one-sample KS test could be used, for example, as a test of normality.

- The one-sample Kolmogorov-Smirnov test statistic $KS_n^{(1)}$ looks at the maximum distance between the empirical distribution function and F_0

$$KS_n^{(1)} = \sup_t \big|\hat{F}_n(t) - F_0(t)\big|$$

- Large values of $KS_n^{(1)}$ provide evidence againts the null hypothesis.

- Under H_0, $\sqrt{n}KS_n^{(1)}$ converges in distribution to a **Kolmogorov distribution** as n goes to infinity.
- The Kolmogorov distribution has the following cumulative distribution function:

$$F_{Kolmo}(t) = 1 - 2\sum_{j=1}^{\infty}(-1)^{(j+1)}e^{-2j^2t^2}$$

- The one-sample KS test can be performed in **R** using the **ks.test** function. For one-sample tests, you have to provide the "name" of the distribution function that you are choosing for F_0.

```r
xx <- rt(100, df=2) ## generate 100 observations from a t-dist with 2 d.f.
ks.test(xx, y="pnorm")  ## test that these data follow Normal(0, 1)
```

```
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  xx
## D = 0.13462, p-value = 0.05333
## alternative hypothesis: two-sided
```

- You can even test that the data follow some other Normal(μ, σ^2) by just providing **mean** and **sd** arguments.

```r
ks.test(xx, y="pnorm", mean=1, sd=2)
```

```
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  xx
## D = 0.29597, p-value = 4.924e-08
## alternative hypothesis: two-sided
```

- Suppose we have data from two groups: $X_1, \ldots, X_n \sim F_X$ and $Y_1, \ldots, Y_m \sim F_Y$. The two-sample KS test performs a test of the following hypothesis

$$H_0 : F_X = F_Y \quad \text{vs.} \quad H_A : F_X \neq F_Y.$$

- In this case, we are only testing whether the distributions of the two groups are different. We are not testing whether observations from group 1 tend to be larger (or smaller) than those from group 2.

- The two-sample KS test compares the empirical distribution functions from these two groups.
- The two-sample KS test statistic is defined as the maximum distance between the two empirical distribution functions:

$$KS_{n,m}^{(2)} = \sup_t \left|\hat{F}_{n,X}(t) - \hat{F}_{m,Y}(t)\right|$$

  Here, $\hat{F}_{n,X}(t) = \frac{1}{n}\sum_{i=1}^{n} I(X_i \leq t)$ and $\hat{F}_{m,Y}(t) = \frac{1}{m}\sum_{j=1}^{m} I(Y_j \leq t)$ denote the empirical distribution functions from the X and Y samples.

- The two-sample KS test statistic also converges to the same limit as the one-sample KS test statistic. In particular, under H_0:

$$\sqrt{\frac{nm}{n+m}} KS_{n,m}^{(2)} \longrightarrow \text{Kolmogorov} \qquad \text{as } n, m \longrightarrow \infty$$

- The **ks.test** function in **R** also performs two-sample KS tests.

```
xx <- rnorm(100)
yy <- rlogis(100)
ks.test(xx, yy)
```

```
##
##  Two-sample Kolmogorov-Smirnov test
##
## data:  xx and yy
## D = 0.2, p-value = 0.03663
## alternative hypothesis: two-sided
```

- We can compute the KS statistic ourselves and check that this matches the value of the KS statistic returned by the **ks.test** function:

```
zz <- c(xx, yy)
zz.order <- sort(zz)
F.x <- ecdf(xx)
F.y <- ecdf(yy)

KS.stat <- max( abs( F.x(zz.order) - F.y(zz.order) ) )
KS.stat
```

```
## [1] 0.2
```

- **Exercise 7.1.** Why does

$$KS^{(2)}_{n,m} = \max_{1 \leq i \leq n+m} \big|\hat{F}_{n,X}(Z_{(i)}) - \hat{F}_{n,Y}(Z_{(i)})\big|,$$

  where $\mathbf{Z} = (Z_1, \ldots, Z_{n+m})$ denotes the pooled sample and $Z_{(1)}, \ldots, Z_{(n+m)}$ denote the order statistics from $\mathbf{Z}$?

7.5 The empirical distribution function and statistical functionals

- In many areas of mathematics, it is common to refer to a function which is a "functions of functions" as a **functional**.
- For example, $T(f)$ which is defined as

$$T(f) = \int_0^1 f^2(x)dx \tag{7.9}$$

  is a functional because $T(f)$ takes arguments which are functions and outputs real numbers.

- Many common parameters that we encounter in statistics can be thought of as functionals where the input of the functional is usually a distribution function.
- For example, the mean is an example of a functional

$$\mu(F) = \int x dF(x) = \int x f(x) dx$$

  As indicated by the notation $\mu(F)$, the value of the mean depends on the underlying distribution function F.
- Also, the variance is an example of a functional

$$\sigma^2(F) = \int (x - \mu(F))^2 dF(x) = \int x^2 dF(x) - \mu^2(F)$$

- The median is an example of a functional

$$\text{med}(F) = F^{-1}(1/2)$$

- The tail probability $P(X_i > c)$ is also a functional

$$\theta_c(F) = \int I(x > c) dF(x)$$

- Many common estimators can be thought of as coming from "plugging in" the empirical cdf into the appropriate statistical functional.
- For example, plugging in the empirical cdf into the mean functional gives:

$$\mu(\hat{F}_n) = \int x d\hat{F}_n(x) = \frac{1}{n}\sum_{i=1}^{n} X_i = \bar{X}$$

- Plugging in the empirical cdf into the tail probability functional gives

$$\theta_c(\hat{F}_n) = \int I(x > c) d\hat{F}_n(x) = \frac{1}{n}\sum_{i=1}^{n} I(X_i > c) = 1 - \hat{F}_n(c)$$

- The sample variance is not quite a plug-in estimate for $\sigma^2(F)$ but it is very close

$$\begin{aligned}\sigma^2(\hat{F}_n) &= \int x^2 d\hat{F}_n(x) - \mu^2(\hat{F}_n) = \frac{1}{n}\sum_{i=1}^{n} X_i^2 - \bar{X}^2 \\ &= \frac{1}{n}\sum_{i=1}^{n}(X_i - \bar{X})^2 = \frac{n-1}{n}\hat{\sigma}^2\end{aligned}$$

- This notation for statistical functionals will be useful when we discuss the bootstrap later in the course.
- The notation for statistical functionals is also very useful in the context of influence functions and robust statistics, but we will not discuss these topics in this course.

7.6 Additional Reading

- Additional reading which covers the material discussed in this chapter includes:
  - Chapter 2 from Wasserman (2006)

Chapter 8

Density Estimation

8.1 Introduction

- In this section, we focus on methods for estimating a **probability density function** (pdf) $f(x)$.

- For a continuous random variable X, areas under the probability density function are probabilities

$$P(a < X < b) = \int_a^b f(x)dx$$

  and $f(x)$ is related to the cumulative distribution function via $f(x) = F'(x)$.

- With parametric approaches to density estimation, you only need to estimate several parameters as these parameters completely determine the form of $f(x)$.

- For example, with a Gaussian distribution you only need to find μ and σ^2 to determine the form of $f(x)$.

- In a nonparametric approach to estimating a pdf, we will assume that our observations $X_1, \ldots, X_n$ are an independent identically distribution sample from a distribution with pdf $f(x)$, but otherwise we will make few assumptions about the particular form of $f(x)$.

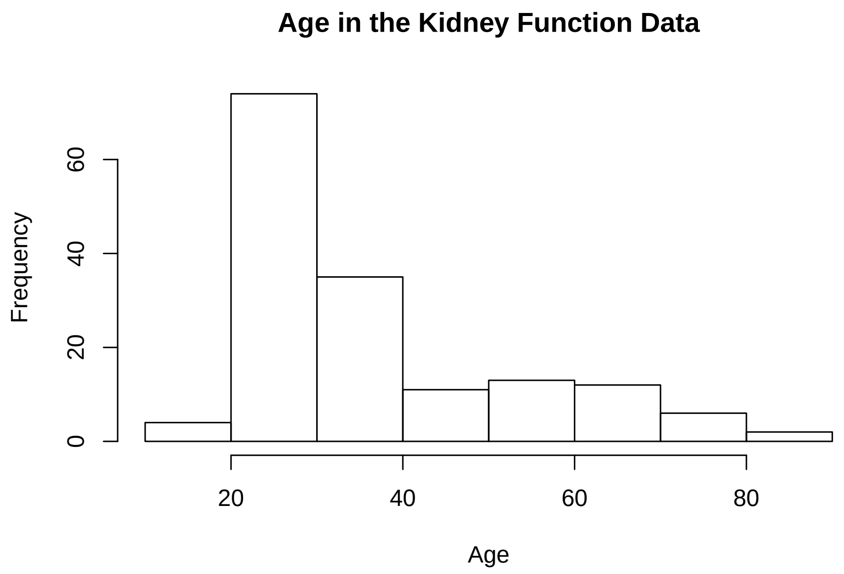


Figure 8.1: Histogram of ages from kidney function data. Data retrieved from: https://web.stanford.edu/~hastie/CASI_files/DATA/kidney.txt

8.2 Histograms

8.2.1 Definition

- While histograms are often thought of as mainly a visualization tool, a histogram can also be thought of as an estimate of the density $f(x)$.
- To construct a histogram, you first need to define a series of "bins": $B_1, \dots, B_{D_n}$.
- Each bin is a left-closed interval. That is, the bins have the form $B_k = [x_0 + (k-1)h_n, x_0 + kh_n)$:

$$\begin{aligned} B_1 &= [x_0, x_0 + h_n) \\ B_2 &= [x_0 + h_n, x_0 + 2h_n) \\ &\vdots \\ B_{D_n} &= [x_0 + (D_n - 1)h_n, x_0 + D_n h_n) \end{aligned}$$

- x_0 - the origin
- h_n - bin width
- D_n - number of bins
- Histograms are based on the counts n_k of observations that fall into each bin:

$$\begin{aligned} n_k &= \#\text{ of observations falling into the } k^{th} \text{ bin} \\ &= \sum_{i=1}^{n} I(x_0 + (k-1)h_n \leq X_i < x_0 + kh_n) \end{aligned}$$

- From the counts n_k, the histogram estimate of the density at a point x in the k^{th} bin (that is if $x_0 + (k-1)h_n \leq x < x_0 + kh_n$), is defined as

$$\hat{f}_{h_n}^{H}(x) = \frac{n_k}{nh_n}$$

- **Note:** Histogram plots often show the actual bin counts n_k rather than the values of $\hat{f}_{h_n}^{H}(x)$.

- To see the motivation for the histogram estimate, notice that if we choose a relatively small value of $h_n > 0$

$$P(a < X_i < a + h_n) = \int_a^{a+h_n} f(t)dt \approx h_n f(c),$$

for any point $a \leq c \leq a + h_n$.

- So, for a point $x \in B_k$, the expected value of $\hat{f}_{h_n}^H(x)$ is

$$
\begin{aligned}
E\{\hat{f}_{h_n}^H(x)\} &= \frac{1}{nh_n} E\{n_k\} \\
&= \frac{1}{nh_n}\sum_{i=1}^{n} P(x_0 + (k-1)h_n \leq X_i < x_0 + kh_n) \\
&= \frac{1}{h_n} P(x_0 + (k-1)h_n \leq X_i < x_0 + kh_n) \\
&\approx f(x)
\end{aligned}
$$

8.2.2 Histograms in R

- In **R**, histograms are computed using the `hist` function

```
hist(x, breaks, probability, plot, ...)
```

- The **breaks** argument
  - Default is "Sturges". This is a method for finding the bin width.
  - Can be a name giving the name of an algorithm for computing bin width (e.g., "Scott" and "FD").
  - Can also be a single number. This gives the number of bins used.
  - Could be a vector giving the breakpoints between bins.
  - Could also be a function which computes the number of bins.

- The **probability** argument. If this is set to FALSE, then the bin counts are shown in the histogram. If set to TRUE, then the bin counts divided by nh_n are shown in the histogram.

- The **plot** argument. If TRUE, a histogram is plotted whenever `hist` is called. If FALSE, a histogram is not plotted when `hist` is called.

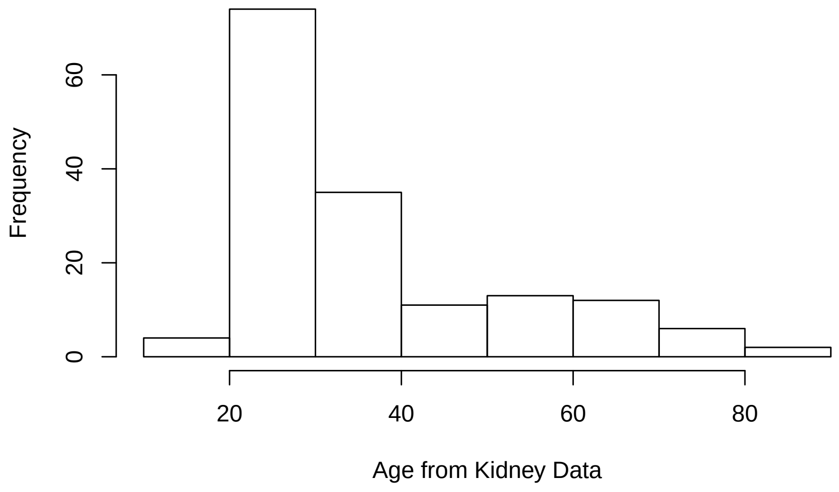
Note: The default for R, is to use right-closed intervals $(a, b]$. This can be changed using the **right** argument of the **hist** function.

- Let's use the kidney function data again to demonstrate the use of histograms in **R**. This time we will focus on the **age** variable.

```
kidney <- read.table("https://web.stanford.edu/~hastie/CASI_files/DATA/kidney.txt",
                     header=TRUE)
```

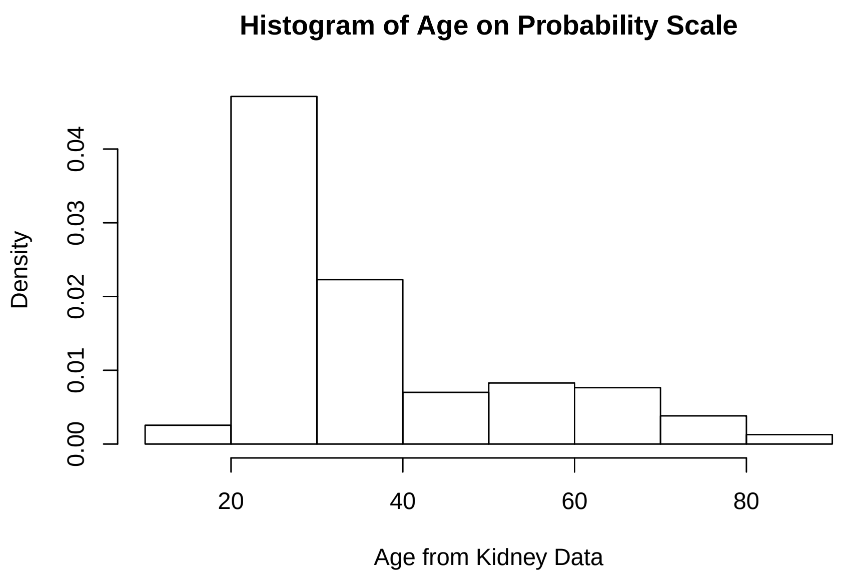
- You can plot a histogram of **age** just by calling the `hist` function.

```
kidney.hist <- hist(kidney$age, main="", xlab="Age from Kidney Data")
```



- Use the `probability = TRUE` argument to plot the density-estimate version of the histogram. This histogram should integrate to 1.

```
kidney.hist2 <- hist(kidney$age, main="Histogram of Age on Probability Scale",
                     xlab="Age from Kidney Data", probability=TRUE)
```



- In addition to generating a histogram plot, the histogram function also returns useful stuff.

```
names(kidney.hist)
```

```
## [1] "breaks"   "counts"   "density"  "mids"     "xname"    "equidist"
```

- **breaks**
  - the boundaries for the histogram bins. The bins are of the form (breaks[k], breaks[k+1]]
- **counts**
  - the number of observations falling into each bin
- **density**
  - the value of the estimated density within each of the bins
- **mids**
  - the midpoint of each of the bins

```
kidney.hist$breaks
```

```
## [1] 10 20 30 40 50 60 70 80 90
```

```
kidney.hist$counts
```

```
## [1]  4 74 35 11 13 12  6  2
```

```
## The following sum should match the first element of kidney.hist$counts[1]
sum(kidney.hist$breaks[1] < kidney$age & kidney$age <= kidney.hist$breaks[2])
```

```
## [1] 4
```

- Let's check that the density values returned by `hist` match our definition of the histogram density estimate in (8.1).

```
binwidth <- kidney.hist$breaks[2] - kidney.hist$breaks[1]
kidney.hist$density
```

```
## [1] 0.002547771 0.047133758 0.022292994 0.007006369 0.008280255 0.007643312
## [7] 0.003821656 0.001273885
```

```
kidney.hist$counts/(length(kidney$age)*binwidth)
```

```
## [1] 0.002547771 0.047133758 0.022292994 0.007006369 0.008280255 0.007643312
## [7] 0.003821656 0.001273885
```

8.2.3 Performance of the Histogram Estimate and Bin Width Selection

8.2.3.1 Bias/Variance Decomposition

- It is common to evaluate the performance of a density estimator through its **mean-squared error** (MSE).

- In general, MSE is a function of bias and variance

$$\text{MSE} = \text{Bias}^2 + \text{Variance}$$

- We will first look at the mean-squared error of $\hat{f}^H_{h_n}(x)$ at a single point x

$$\begin{aligned}
\text{MSE}\{\hat{f}^H_{h_n}(x)\} &= E\Big(\{\hat{f}^H_{h_n}(x) - f(x)\}^2\Big) \\
&= E\Big(\Big[\hat{f}^H_{h_n}(x) - E\{\hat{f}^H_n(x)\} + E\{\hat{f}^H_{h_n}(x)\} - f(x)\Big]^2\Big) \\
&= E\Big(\Big[\hat{f}^H_{h_n}(x) - E\{\hat{f}^H_n(x)\}\Big]^2\Big) + E\Big(\Big[E\{\hat{f}^H_{h_n}(x)\} - f(x)\Big]^2\Big) \\
&+ \;2E\Big(\Big[\hat{f}^H_{h_n}(x) - E\{\hat{f}^H_n(x)\}\Big]\Big[E\{\hat{f}^H_{h_n}(x)\} - f(x)\Big]\Big) \\
&= \underbrace{\text{Var}\{\hat{f}^H_{h_n}(x)\}}_{\text{Variance}} + \underbrace{\Big(E\{\hat{f}^H_{h_n}(x)\} - f(x)\Big)^2}_{\text{Bias Squared}}
\end{aligned}$$

- In general, as the bin width h_n increases, the histogram estimate will have less variation but will become more biased.

8.2.3.2 Bias and Variance of the Histogram Estimate

- Recall that, for a histogram estimate, we have D_n bins where the k^{th} bin takes the form
$$B_k = [x_0 + (k-1)h_n, x_0 + kh_n)$$
- For a point $x \in B_k$, that "belongs" to the k^{th} bin, the histogram density estimate is
$$\hat{f}^H_n(x) = \frac{n_k}{nh_n}, \qquad \text{where } n_k = \text{ number of observations falling into bin } B_k$$
- To better examine what happens as n changes, we will define the function $A_{h_n,x_0}(x)$ as the function which returns the index of the interval to which x belongs.
- For example, if $x_0 = 0$, $h_n = 1/3$, and $x = 1/2$, then $A_{h_n,x_0}(x) = 2$.
- So, we can also write the histogram density estimate at the value x as
$$\hat{f}^H_{h_n}(x) = \frac{n_{A_{h_n,x_0}(x)}}{nh_n}$$

- **Exercise 8.1.** Suppose $x_0 = -2$ and $h_n = 1/2$. What are the values of $A_{h_n,x_0}(-1)$, $A_{h_n,x_0}(1.3)$, and $A_{h_n,x_0}(0.75)$?

- Note that we can express $n_{A_{h_n,x_0}(x)}$ as

$$n_{A_{h_n,x_0}(x)} = \sum_{i=1}^{n} I\Big(X_i \in B_{A_{h_n,x_0}(x)}\Big)$$

- Hence, is a binomial random variable with n trials and success probability $p_{h_n,x_0}(x)$ (why?)

$$n_{A_{h_n}(x)} \sim \text{Binomial}\{n, p_{h_n,x_0}(x)\}$$

- The success probability $p_{h_n,x_0}(x)$ is defined as

$$\begin{aligned} p_{h_n,x_0}(x) &= P\Big\{X_i \text{ falls into bin } B_{A_{h_n,x_0}}(x)\Big\} \\ &= \int_{x_0+(A_{h_n,x_0}(x)-1)h_n}^{x_0+A_{h_n,x_0}(x)h_n} f(t)dt. \end{aligned}$$

- Because $n_{A_{h_n,x_0}(x)}$ follows a binomial distribution, we know that

$$\begin{aligned} E(n_{A_{h_n,x_0}(x)}) &= np_{h_n,x_0}(x) \\ \text{Var}(n_{A_{h_n,x_0}(x)}) &= np_{h_n,x_0}(x)\{1-p_{h_n,x_0}(x)\} \end{aligned}$$

- So, we can express the bias of the histogram density estimate $\hat{f}_{h_n}^H(x) = n_{A_{h_n,x_0}(x)}/(nh_n)$ as

$$\begin{aligned} \text{Bias}\{\hat{f}_{h_n}^H(x)\} &= E\{\hat{f}_{h_n}^H(x)\} - f(x) \\ &= \frac{1}{nh_n}E(n_{A_{h_n,x_0}(x)}) - f(x) \\ &= \frac{p_{h_n,x_0}(x)}{h_n} - f(x), \end{aligned}$$

  and we can express the variance as:

$$\begin{aligned} \text{Var}\{\hat{f}_{h_n}^H(x)\} &= \frac{1}{n^2h_n^2}\text{Var}(n_{A_{h_n,x_0}(x)}) \\ &= \frac{p_{h_n,x_0}(x)\{1-p_{h_n,x_0}(x)\}}{nh_n^2} \end{aligned}$$

- Using the approximation $f(t) \approx f(x) + f'(x)(t-x)$ for t close to x, we have that

$$\begin{aligned}
\frac{p_{h_n,x_0}(x)}{h_n} &= \frac{1}{h_n}\int_{x_0+(A_{h_n,x_0}(x)-1)h_n}^{x_0+A_{h_n,x_0}(x)h_n} f(t)dt \\
&\approx \frac{1}{h_n}\int_{x_0+(A_{h_n,x_0}(x)-1)h_n}^{x_0+A_{h_n,x_0}(x)h_n} f(x)dt + \frac{f'(x)}{h_n}\int_{x_0+(A_{h_n}(x)-1)h_n}^{x_0+A_{h_n,x_0}(x)h_n}(t-x)dt \\
&= f(x) + \frac{f'(x)}{2h_n}\Big[(t-x)^2\Big|_{x_0+(A_{h_n,x_0}(x)-1)h_n}^{x_0+A_{h_n,x_0}(x)h_n}\Big] \\
&= f(x) + \frac{f'(x)}{2h_n}\Big[(x_0+A_{h_n,x_0}(x)h_n)^2 - (x_0+(A_{h_n,x_0}(x)-1)h_n)^2 - 2xh_n\Big] \\
&= f(x) + \frac{f'(x)}{2h_n}\Big[2x_0A_{h_n,x_0}(x)h_n + A^2_{h_n,x_0}(x)h_n^2 - 2x_0(A_{h_n,x_0}(x)-1)h_n \\
&\quad -(A_{h_n,x_0}(x)-1)^2h_n^2 - 2xh_n\Big] \\
&= f(x) + \frac{f'(x)}{2h_n}\Big[2x_0h_n + 2A_{h_n,x_0}(x)h_n^2 - h_n^2 - 2xh_n\Big] \\
&= f(x) + f'(x)\Big[h_n/2 - [x - x_0 - \{A_{h_n,x_0}(x)-1\}h_n]\Big]
\end{aligned}$$

- So, the bias of the histogram density estimate $\hat{f}^H_{h_n}(x)$ is

$$\begin{aligned}
\text{Bias}\{\hat{f}^H_{h_n}(x)\} &= \frac{p_{h_n,x_0}(x)}{h_n} - f(x) \\
&\approx f'(x)\Big[h_n/2 - [x - x_0 - \{A_{h_n,x_0}(x)-1\}h_n]\Big] \quad (8.1)
\end{aligned}$$

- Choosing a very small bin width h_n will result in a small bias because the left endpoint of the bin $x_0 + (A_{h_n}(x)-1)h_n$ will always be very close to x.

- Now, let us turn to the variance of the histogram estimate at x:

$$\begin{aligned}
\text{Var}\{\hat{f}^H_{h_n}(x)\} &= \frac{p_{h_n,x_0}(x)}{nh_n^2}\{1 - p_{h_n,x_0}(x)\} \\
&\approx \frac{f(x) + f'(x)\{\frac{h_n}{2} - [x - x_0 - (A_{h_n,x_0}(x)-1)h_n]\}}{nh_n}\{1 - p_{h_n,x_0}(x)\} \\
&\approx \frac{f(x)}{nh_n} \quad (8.2)
\end{aligned}$$

- For a more detailed description of the above approximation see Scott (1979) or Chapter 6 of Wasserman (2006).
- Note that large bin widths will reduce the variance of $\hat{f}^H_{h_n}(x)$.

8.2.3.3 Pointwise Approximate Mean Squared Error

- Using (8.2) and (8.1), the approximate mean-squared error (AMSE) of the histogram density estimate at a particular point x is given by

$$\begin{aligned}
\text{MSE}\{\hat{f}^H_{h_n}(x)\} &= E\Big(\{\hat{f}^H_{h_n}(x) - f(x)\}^2\Big) \\
&= \Big(\text{Bias}\{\hat{f}^H_{h_n}(x)\}\Big)^2 + \text{Var}\{\hat{f}^H_{h_n}(x)\} \\
&\approx \frac{h_n^2[f'(x)]^2}{4} - h_n f'(x)[x - x_0 - \{A_{h_n,x_0}(x) - 1\}h_n] \\
&+ [f'(x)]^2\Big(x - x_0 - \{A_{h_n,x_0}(x) - 1\}h_n\Big)^2 + \frac{f(x)}{nh_n} \\
&= \text{AMSE}\{\hat{f}^H_{h_n}(x)\} \qquad (8.3)
\end{aligned}$$

- For any approach to bin width selection, we should have $h_n \longrightarrow 0$ and $nh_n \longrightarrow \infty$.
- This MSE approximation depends on a particular choice of x.
- Difficult to use (8.3) as a criterion for selecting the bandwidth because the best choice of h_n will usually depend on your choice of x.

8.2.3.4 Integrated Mean Squared Error and Optimal Histogram Bin Width

- Using the approximate mean integrated squared error (AMISE) allows us to find an optimal bin width that does not depend on a particular choice of x.
- The AMISE is defined as

$$\begin{aligned}
\text{AMISE}\{\hat{f}^H_{h_n}\} &= E\Big\{\int_{-\infty}^{\infty} \{\hat{f}^H_{h_n}(x) - f(x)\}^2 dx\Big\} \\
&= \int_{-\infty}^{\infty} \text{MSE}\{\hat{f}^H_{h_n}(x)\} dx
\end{aligned}$$

Using the previously derived approximation (8.3) for the AMSE, it can be shown that

$$\text{MISE}\{\hat{f}^H_{h_n}\} \approx \frac{1}{nh_n} + \frac{h_n^2}{12}\int_{-\infty}^{\infty} [f'(x)]^2 dx \qquad (8.4)$$

- To select the optimal bin width, we minimize the MISE as a function of h_n.
- Minimizing (8.4), as a function of h_n yields the following formula for the optimal bin width

$$h_n^{opt} = \Big(\frac{6}{n \int_{-\infty}^{\infty} [f'(x)]^2 dx} \Big)^{1/3} = C n^{-1/3} \tag{8.5}$$

- Notice that $h_n^{opt} \longrightarrow 0$ and $n h_n^{opt} \longrightarrow \infty$ as $n \longrightarrow \infty$.
- Notice also that the optimal bin width depends on the unknown quantity $\int_{-\infty}^{\infty} [f'(x)]^2 dx$.

8.2.4 Choosing the Histogram Bin Width

- We will mention three rules for selecting the bin width of a histogram.
  - Scott rule: (based on the optimal bin width formula (8.5))
  - Friedman and Diaconis (FD) rule (also based on the optimal bin width formula (8.5))
  - Sturges rule: (based on wanting the histogram to look Gaussian when the data are in fact Gaussian-distributed)

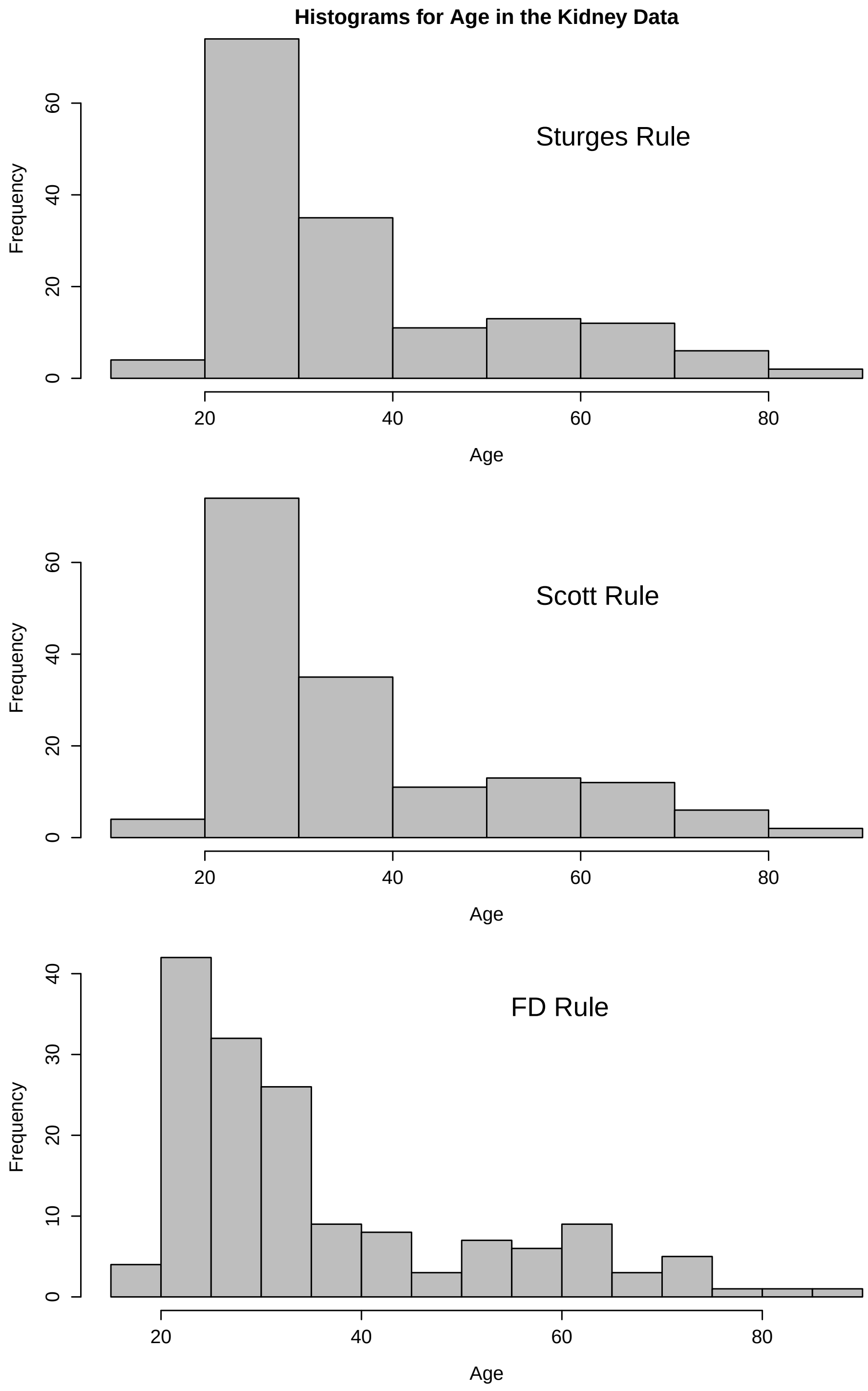
- Both the Scott and the FD rule are based on the optimal bin width formula (8.5).
- The main problem with using the formula (8.5) is the presence of $\int_{-\infty}^{\infty} [f'(x)]^2 dx$.
- **Solution:** See what this quantity looks like if we assume that $f(x)$ corresponds to a Normal(μ, σ^2) density.
- With the assumption that $f(x) = \text{Normal}(\mu, \sigma^2)$:

$$h_n^{opt} = 3.5 \sigma n^{-1/3}$$

- **Scott rule**: Use $h_n^* = 3.5 \hat{\sigma} n^{-1/3}$, where $\hat{\sigma}$ denotes the sample standard deviation.

- **FD rule**: Use $h_n^* = 2 \times IQR \times n^{-1/3}$. This is a somewhat more robust choice of h_n as it is not as sensitive to outliers.

- **Sturges rule**: The bin width is chosen so that we have $1 + log_2(n)$ bins. This choice tends to give wide intervals.


Histograms for Age in the Kidney Data
Sturges Rule
Scott Rule
FD Rule
Frequency
Age
0
20
40
60
80
10
30

8.3 A Box-type Density Estimate

- A related estimator $\hat{f}^B_{h_n}$ of the density $f(x)$ uses a "sliding bin" at each point x to calculate the estimate of $f(x)$.

- Specifically, the "box estimate" $\hat{f}^B_{h_n}(x)$ at the point x is defined as

$$\begin{aligned}\hat{f}^B_{h_n}(x) &= \frac{1}{2nh_n}\Big[\#\text{ of observations falling in the interval } (x-h_n, x+h_n)\Big] \\ &= \frac{1}{2nh_n}\sum_{i=1}^{n} I(x-h_n < X_i < x+h_n)\end{aligned}$$

- In other words, for each x we are forming a bin of width $2h_n$ around x, and we are counting the number of observations that fall in this bin.

- You can think of each point x as being the center of its own bin.

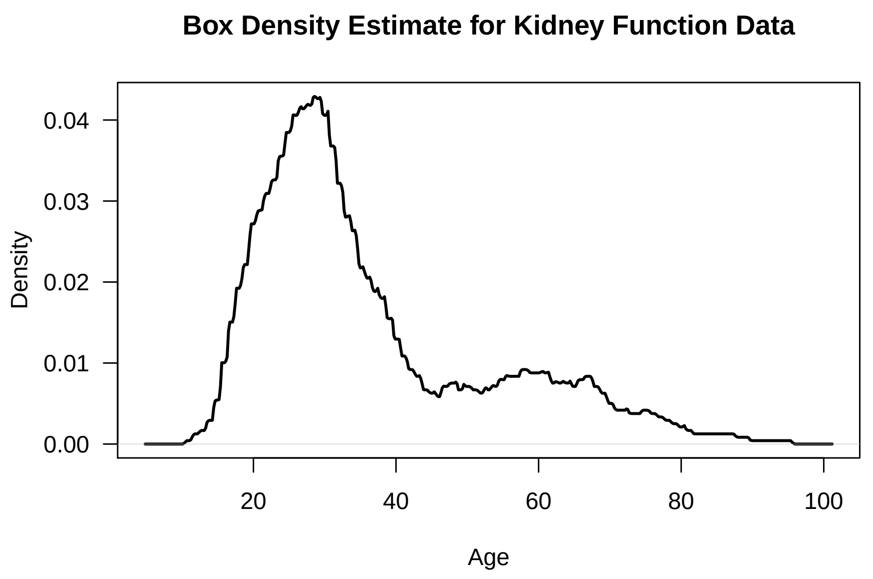
- The expectation of the box estimator at the point x is

$$E\{\hat{f}^B_{h_n}(x)\} = \frac{1}{2h_n}P(x-h_n < X_i < x+h_n) \approx f(x)$$

- Unlike the histogram, the box estimate does not require the density estimate to be constant within each bin.

- Also, histograms can have dramatic changes near the bin edges while the box estimate suffers less from this problem.

- However, plots of the box estimate will still largely be non-smooth and have a "jagged" appearance.

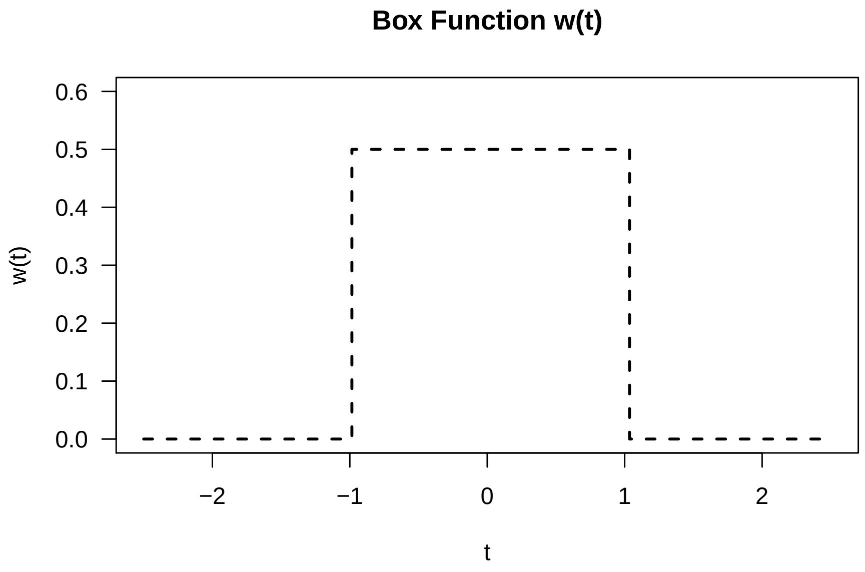


- We can also express $\hat{f}^B_{h_n}(x)$ in the following way:

$$\hat{f}^B_{h_n}(x) = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{h_n}w\Big(\frac{X_i - x}{h_n}\Big), \tag{8.6}$$

where $w(t)$ is defined as the following "box function"

$$w(t) = \begin{cases} \frac{1}{2} & \text{if } |t| < 1 \\ 0 & \text{otherwise} \end{cases}$$



- While the estimator $\hat{f}_{h_n}^B$ does seem reasonable, it always results in density estimates which are not "smooth."

- Most kernel density estimates are formed by replacing the box function $w(t)$ with a smoother function.

- **Exercise 8.2**. Write an **R** function which computes the $\hat{f}_{h_n}^B(x)$ at a collection of specified points.

- **Exercise 8.3**. Suppose we have observations $(X_1, X_2, X_3, X_4) = (-1, 0, 1/2, 1)$. Assuming $h_n = 1/2$, plot $w(\frac{X_i - x}{h_n})/nh_n$ for $i = 1, \dots, 4$ and plot the box density estimate $\hat{f}_{h_n}^B(x)$.

- **Exercise 8.4**. Suppose we have i.i.d. observations $X_1, \dots, X_n \sim F$ where the cdf F is assumed to be continuous. What is $\mathrm{Var}\{\hat{f}_{h_n}^B(x)\}$?

8.4 Kernel Density Estimation

8.4.1 Definition

- Kernel density estimates are a generalization of the box-type density estimate $\hat{f}_{h_n}^{B}(x)$.

- With kernel density estimation, we replace the "box function" in (8.6) with a function $K(\cdot)$ which is much smoother.

- The function $K(\cdot)$ will also give higher weight to observations which are closer to x than those that are further away from x.

- A kernel density estimator $\hat{f}_{h_n}(x)$ of the density $f(x)$ is defined as

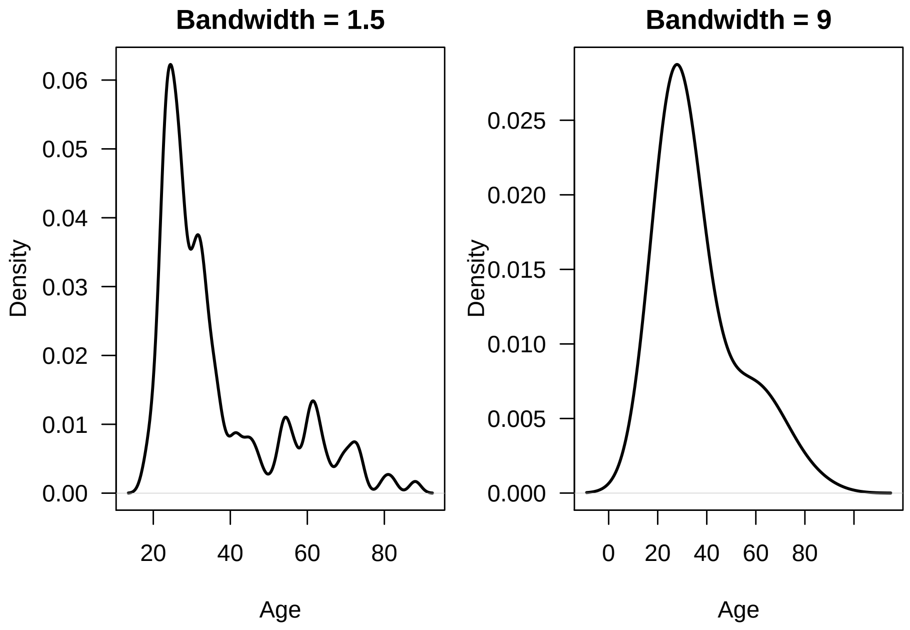
$$\hat{f}_{h_n}(x) = \frac{1}{nh_n}\sum_{i=1}^{n} K\Big(\frac{x - X_i}{h_n}\Big) \tag{8.7}$$

- The function $K(\cdot)$ is referred to as the **kernel function**.

- The scalar term $h_n > 0$ is called the **bandwidth**.

- The value of the bandwidth h_n largely determines how "bumpy" the density estimate will appear.

- The appearance and the statistical performance of $\hat{f}_{h_n}(x)$ depend much more on the value of h_n than on the choice of kernel function.



- Kernel functions are usually chosen so that

$$K(t) \geq 0 \quad \text{for all } t$$

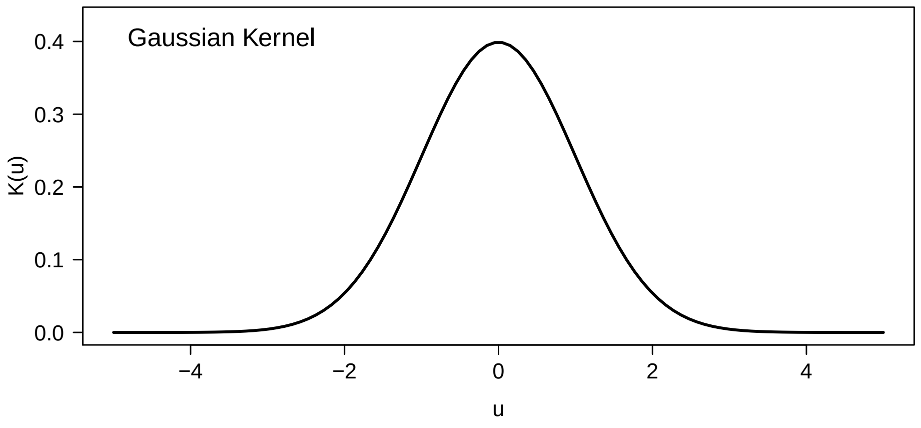
and that they also satisfy the following properties:

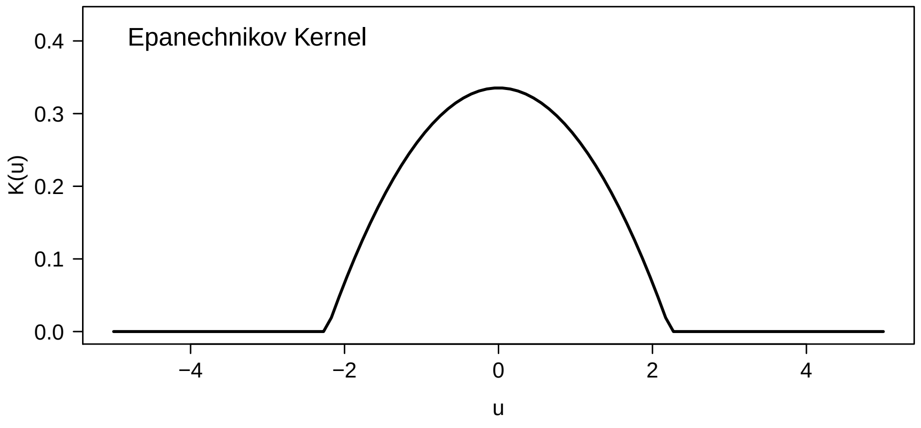
$$K(t) = K(-t) \qquad \int_{-\infty}^{\infty} K(t)dt = 1 \qquad \int_{-\infty}^{\infty} K^2(t)dt < \infty$$

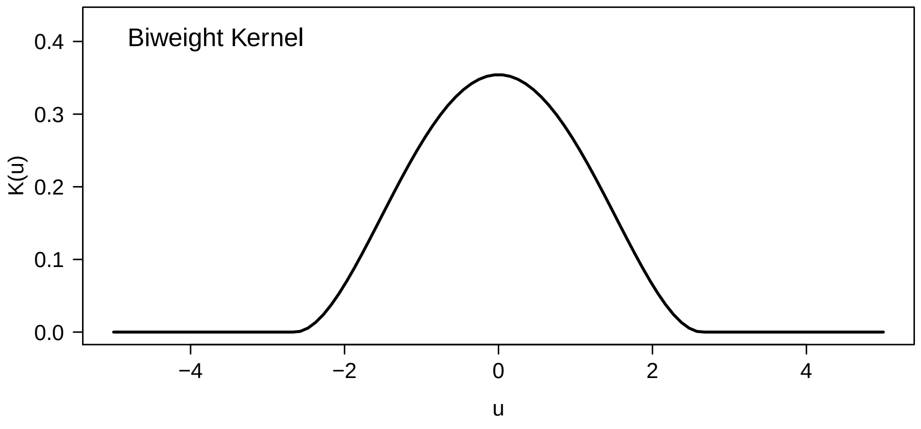
- You can think of $K(u)$ as a probability density function which is symmetric around 0.

- Some of the most common choices of kernel functions include

$$\begin{aligned} \text{Gaussian :} \quad & K(u) = \exp(-u^2/2)/\sqrt{2\pi} \\ \text{Epanechnikov :} \quad & K(u) = \tfrac{3}{4\sqrt{5}}(1 - \tfrac{1}{5}u^2)I(|u| < \sqrt{5}) \\ \text{biweight :} \quad & K(u) = \tfrac{15}{16\sqrt{7}}(1 - \tfrac{1}{7}u^2)^2 I(|u| < \sqrt{7}) \end{aligned}$$

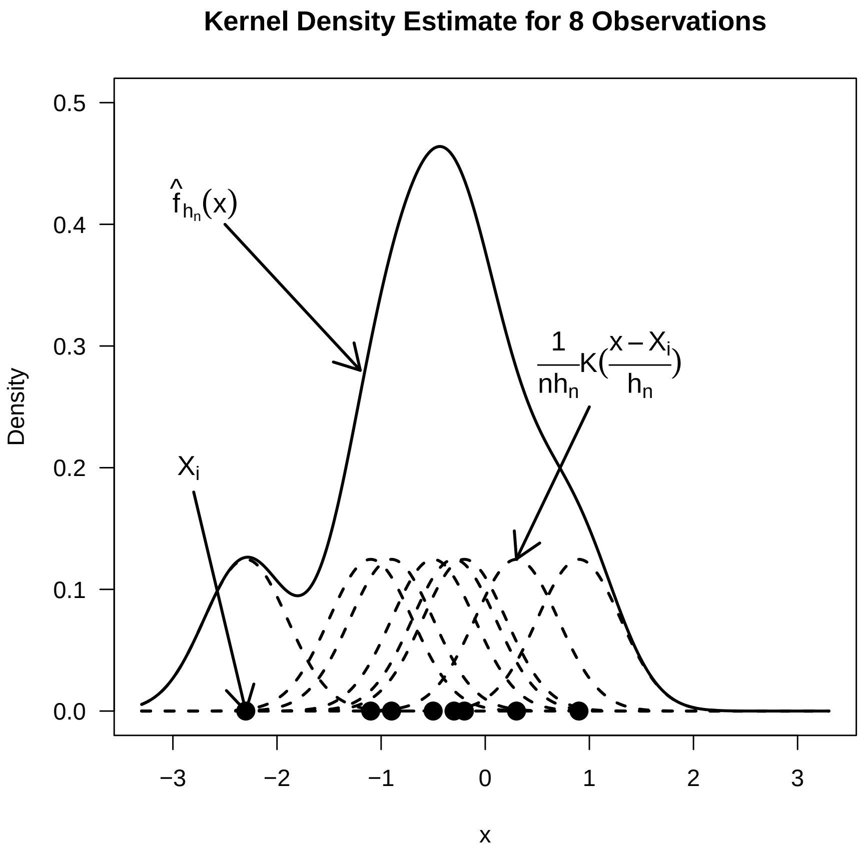






- When plotting $\frac{1}{nh_n}K(\frac{x-X_i}{h_n})$ as a function of x, it should look like a "small hill" centered around X_i.
- As h_n decreases, $\frac{1}{nh_n}K(\frac{x-X_i}{h_n})$ becomes more strongly concentrated around X_i and has a higher peak.

- The kernel density estimate $\hat{f}_{h_n}(x)$ is a sum of all these "small hills".



- The nice thing about (8.7) is that it guarantees that $\hat{f}_{h_n}(x)$ is a probability density function

$$\begin{aligned}\int_{-\infty}^{\infty} \hat{f}_{h_n}(x)dx &= \int_{-\infty}^{\infty} \frac{1}{nh_n}\sum_{i=1}^{n} K\Big(\frac{x-X_i}{h_n}\Big)dx \\ &= \frac{1}{n}\sum_{i=1}^{n}\int_{-\infty}^{\infty}\frac{1}{h_n}K\Big(\frac{x-X_i}{h_n}\Big)dx \\ &= 1\end{aligned}$$

- Also, formula (8.7) guarantees that $\hat{f}_{h_n}(x)$ inherits the smoothness properties of $K(u)$

$$\hat{f}'_{h_n}(x) = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{h_n^2}K'\Big(\frac{x - X_i}{h_n}\Big)$$

8.4.2 Bias, Variance, and AMISE of Kernel Density Estimates

- As with the bin width in histogram estimation, the bias/variance tradeoff drives the best choice of the bandwidth h_n in kernel density estimation.

Approximate Bias

- The exact expectation of a kernel density estimate $\hat{f}_{h_n}(x)$ is

$$\begin{aligned} E\{\hat{f}_{h_n}(x)\} &= \frac{1}{nh_n}\sum_{i=1}^{n}E\Big\{K\Big(\frac{x - X_i}{h_n}\Big)\Big\} \\ &= \frac{1}{h_n}E\Big\{K\Big(\frac{x - X_1}{h_n}\Big)\Big\} \\ &= \frac{1}{h_n}\int_{-\infty}^{\infty}K\Big(\frac{x - t}{h_n}\Big)f(t)dt \\ &= \int_{-\infty}^{\infty}K(u)f(x - uh_n)du \end{aligned}$$

  Above, we used the substitution $u = (x - t)/h_n$.

- Thus, the exact bias of $\hat{f}_{h_n}(x)$ is

$$\begin{aligned} \text{Bias}\{\hat{f}_{h_n}(x)\} &= \int_{-\infty}^{\infty}K(u)f(x - uh_n)du - f(x) \\ &= \int_{-\infty}^{\infty}K(u)\{f(x - uh_n) - f(x)\}du \qquad (8.8) \end{aligned}$$

- The expression for bias shown in (8.8) will only have a closed form for special choices of $K(\cdot)$ and $f(\cdot)$.

- Nevertheless, we can get a reasonable approximation of this bias for small h_n.

- To approximate the bias for small h_n, we can use the following Taylor series approximation

$$f(x - uh_n) - f(x) \approx -uh_n f'(x) + \frac{u^2 h_n^2}{2} f''(x)$$

Plugging this approximation into our expression for the bias in (8.8) gives:

$$\begin{aligned}
\text{Bias}\{\hat{f}_{h_n}(x)\} &\approx -h_n f'(x) \int_{-\infty}^{\infty} uK(u)du + \frac{h_n^2 f''(x)}{2} \int_{-\infty}^{\infty} u^2 K(u)du \\
&= \frac{h_n^2 f''(x)}{2} \int_{-\infty}^{\infty} u^2 K(u)du \\
&= \frac{h_n^2 f''(x)\mu_2(K)}{2} \qquad \text{where } \mu_2(K) = \int_{-\infty}^{\infty} u^2 K(u)du \\
&= \text{ABias}\{\hat{f}_{h_n}(x)\} \qquad (8.9)
\end{aligned}$$

Here, ABias$\{\hat{f}_{h_n}(x)\}$ stands for the approximate bias.

- Formula (8.9) shows the direct dependence of the magnitude of bias on the value of the bandwidth.

- Formula (8.9) also shows the effect of the curvature $f''(x)$ of the density on the magnitude of bias.

Approximate Variance

- The exact variance of a kernel density estimate $\hat{f}_{h_n}(x)$ is

$$\begin{aligned}
\text{Var}\{\hat{f}_{h_n}(x)\} &= \frac{1}{n^2 h_n^2} \sum_{i=1}^{n} \text{Var}\Big\{K\Big(\frac{x - X_i}{h_n}\Big)\Big\} \\
&= \frac{1}{nh_n^2} \text{Var}\Big\{K\Big(\frac{x - X_1}{h_n}\Big)\Big\} \\
&= \frac{1}{nh_n^2} \int_{-\infty}^{\infty} K^2\Big(\frac{x - t}{h_n}\Big) f(t)dt - \frac{1}{n}\Big[\frac{1}{h_n} E\Big\{K\Big(\frac{x - X_1}{h_n}\Big)\Big\}\Big]^2 \\
&= \frac{1}{nh_n} \int_{-\infty}^{\infty} K^2(u) f(x - uh_n)du - \frac{1}{n}\Big[\text{Bias}\{\hat{f}_{h_n}(x)\} + f(x)\Big]^2
\end{aligned}$$

- We will ignore the last term because it is of order $1/n$. Then, if we use the Taylor series approximation $f(x - uh_n) = f(x) - uh_n f'(x) + ...$, we have:

$$\begin{aligned}
\text{Var}\{\hat{f}_{h_n}(x)\} &\approx \frac{f(x)}{nh_n}\int_{-\infty}^{\infty} K^2(u)du - \frac{f'(x)}{n}\int_{-\infty}^{\infty} uK^2(u)du + ... \\
&\approx \frac{f(x)\kappa_2(K)}{nh_n} \qquad \text{where } \kappa_2(K) = \int_{-\infty}^{\infty} K^2(u)du \\
&= \text{AVar}\{\hat{f}_{h_n}(x)\}
\end{aligned}$$

Approximate Mean Integrated Squared Error (AMISE)

- Using our approximations for the bias and variance, we can get an approximate expression for the mean-squared error of $\hat{f}_{h_n}(x)$ at the point x

$$\begin{aligned}
\text{MSE}\{\hat{f}_{h_n}(x)\} &\approx \text{AVar}\{\hat{f}_{h_n}(x)\} + \Big(\text{ABias}\{\hat{f}_{h_n}(x)\}\Big)^2 \\
&= \frac{f(x)\kappa_2(K)}{nh_n} + \frac{h_n^4[f''(x)]^2\mu_2^2(K)}{4} \qquad (8.10)
\end{aligned}$$

- Notice that if we want the MSE to go to 0 as $n \longrightarrow \infty$, we need the following two things to happen:
  - $h_n \longrightarrow 0$ as $n \longrightarrow \infty$
  - $nh_n \longrightarrow \infty$ as $n \longrightarrow \infty$

- The approximate mean integrated squared error (AMISE) of the kernel density estimator is obtained by integrating the approximation (8.10) for MSE across x

$$\begin{aligned}
\text{AMISE}\{\hat{f}_{h_n}\} &= \frac{\kappa_2(K)}{nh_n}\int_{-\infty}^{\infty} f(x)dx + \frac{h_n^4\mu_2^2(K)}{4}\int_{-\infty}^{\infty}[f''(x)]^2dx \\
&= \frac{\kappa_2(K)}{nh_n} + \frac{h_n^4\mu_2^2(K)}{4}\int_{-\infty}^{\infty}[f''(x)]^2dx \qquad (8.11)
\end{aligned}$$

8.4.3 Bandwidth Selection with the Normal Reference Rule and Silverman's "Rule of Thumb"

- If we differentiate the formula for AMISE given in (8.11) with respect to h_n and set it to zero, we get the following equation for h_n^{opt} which would be the bandwidth minimizing AMISE$\{\hat{f}_{h_n}\}$:

$$0 = \frac{-\kappa_2(K)}{n(h_n^{opt})^2} + (h_n^{opt})^3\mu_2^2(K)\int_{-\infty}^{\infty}[f''(x)]^2dx$$

- The solution of the above equation gives the optimal bandwidth:

$$h_n^{opt} = n^{-1/5}\Big(\int_{-\infty}^{\infty}[f''(x)]^2dx\Big)^{-1/5}\kappa_2(K)^{1/5}\mu_2(K)^{-2/5}$$

- For the case of a Gaussian kernel,

$$\begin{aligned}\kappa_2(K) &= \frac{1}{2\pi}\int_{-\infty}^{\infty}e^{-u^2}du = \frac{1}{2\sqrt{\pi}}\int_{-\infty}^{\infty}\frac{\sqrt{2}}{\sqrt{2\pi}}e^{-u^2}du = \frac{1}{2\sqrt{\pi}}\\ \mu_2(K) &= \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{\infty}u^2e^{-u^2/2}du = 1\end{aligned}$$

  so that, in the case of a Gaussian kernel, the optimal bandwidth is given by

$$h_n^{opt} = n^{-1/5}(2\sqrt{\pi})^{-1/5}\Big(\int_{-\infty}^{\infty}[f''(x)]^2dx\Big)^{-1/5}$$

- However, the optimal bandwidth h_n^{opt} for a kernel density estimate depends on the unknown quantity

$$\int_{-\infty}^{\infty}[f''(x)]^2dx$$

- Similar to the Scott and FD rules for choosing the bin width of histogram, one way of setting the bandwidth h_n of a kernel density estimate is to use the value of $\int_{-\infty}^{\infty}[f''(x)]^2dx$ when it is assumed that $f(x)$ is the density of a Normal$(0,\sigma^2)$ random variable.

- If $f(x) = \text{Normal}(0,\sigma^2)$, then

$$\int_{-\infty}^{\infty}[f''(x)]^2dx = \frac{3}{8\sqrt{\pi}\sigma^5}$$

- If we use this assumption about $f''(x)$ and assume a Gaussian kernel, the formula for the optimal bandwidth becomes bandwidth is

$$h_n^{opt} = n^{-1/5}(2\sqrt{\pi})^{-1/5}\Big(\frac{3}{8\sqrt{\pi}\sigma^5}\Big)^{-1/5} = \sigma n^{-1/5}\sigma\Big(\frac{4}{3}\Big)^{1/5} \approx 1.06\sigma n^{-1/5}$$

- The rule $h_n^{opt} = 1.06\sigma n^{-1/5}$ works pretty well if the true density has a roughly Gaussian shape. For example, unimodal, non-heavy tails, not too strong skewness, etc.

- The **Normal reference rule** for the bandwidth is

$$h_n^{NR} = 1.06\tilde{\sigma} n^{-1/5}$$

- Here, $\tilde{\sigma}$ is usually either $\tilde{\sigma} = \hat{\sigma}$ or $\tilde{\sigma} = s$.
- $h_n = 1.06\hat{\sigma} n^{-1/5}$ is typically too large if $f(x)$ is multimodal or if $f(x)$ has substantial skewness.
- This oversmoothing effect of $h_n = 1.06\hat{\sigma} n^{-1/5}$ can be reduced somewhat by replacing $\hat{\sigma}$ with

$$s = \min\Big\{ \hat{\sigma}, \frac{IQR}{1.34} \Big\}$$

- For multimodal distributions, we typically have $\hat{\sigma} \leq IQR/1.34$ while for strongly skewed distributions such as the log-normal distribution we typically have $\hat{\sigma} \geq IQR/1.34$.
- For these reasons, $\tilde{\sigma} = s$ is often suggested when using the Normal reference rule.

- Silverman's "rule-of-thumb" for the bandwidth is

$$h_n^{SR} = 0.9 s n^{-1/5}$$

(see Chapter 3 of Silverman (2018)).

- This rule is just an adjustment for the fact that $1.06 s n^{-1/5}$ is still too large for many skewed or multimodal distribution and using 0.9 instead of 1.06 reduces this problem.

- **Exercise 8.5** Assuming that $f(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-x^2/2\sigma^2}$, show that

$$\int_{-\infty}^{\infty} [f''(x)]^2 dx = \frac{3}{8\sqrt{\pi}\sigma^5}$$

- **Exercise 8.6** Suppose $X_1, \ldots, X_n \sim N(0, \sigma^2)$. Suppose that we have density estimate $\hat{f}_{h_n}(x)$ that uses the Gaussian kernel.
  - Compute the exact values of $E\{\hat{f}_{h_n}(x)\}$ and $\text{Var}\{\hat{f}_{h_n}(x)\}$.
  - Compute AMISE$\{\hat{f}_{h_n}\}$. Can you get an expression for the value of h_n which minimizes AMISE?

8.5 Cross-Validation for Bandwidth Selection

8.5.1 Squared-Error Cross-Validation

- The usual goal in bandwidth selection is to choose the bandwidth which minimizes the mean integrated squared error (MISE)

$$\text{MISE}(h) = E\Big[\int \{\hat{f}_h(x) - f(x)\}^2 dx\Big] \tag{8.12}$$

  or some related criterion which also measures an expected discrepancy between $\hat{f}_h(x)$ and $f(x)$.

- The MISE can be rewritten as

$$\text{MISE}(h) = E\Big[\int \hat{f}_h^2(x)dx\Big] - 2\int E\Big[\hat{f}_h(x)\Big]f(x)dx + \int f^2(x)dx \tag{8.13}$$

- If the goal is to minimize MISE(h), we can ignore the last term since it does not depend on h.

- Our goal will be to find a bandwidth that minimizes an estimate of MISE(h) (an estimate which ignores the last term in (8.13)), and this estimate will use a technique called leave-one-out **cross-validation**.

- Regarding the first term in (8.13), we can just estimate this expectation with

$$\int \hat{f}_h^2(x)dx$$

- The second term in (8.13) is trickier because it has $f(x)$ in it.

- We can simplify this term though

$$\int E\Big[\hat{f}_h(x)\Big]f(x)dx = E\Bigg[\frac{1}{h}K\Big(\frac{X_1 - X_2}{h}\Big)\Bigg]$$

  (why is this true?)

- We can construct an estimate of (8.13) by first defining the following "leave-one-out" estimate

$$\hat{f}_{h,-i}(x) = \frac{1}{(n-1)h}\sum_{j \neq i} K\Big(\frac{x - X_j}{h}\Big)$$

- In other words, $\hat{f}_{h,-i}(x)$ is a density estimate constructed from using bandwidth h and all the data except for the i^{th} observation.
- Then, we are going use the following quantity to estimate $\int E[\hat{f}_h(x)]f(x)dx$

$$\frac{1}{n}\sum_{i=1}^{n}\hat{f}_{h,-i}(X_i) = \frac{1}{n(n-1)}\sum_{i=1}^{n}\sum_{j\neq i}\frac{1}{h}K\Big(\frac{X_i - X_j}{h}\Big)$$

- The term $\frac{1}{n}\sum_{i=1}^{n}\hat{f}_{h,-i}(X_i)$ is referred to as a leave-one-out cross-validation estimate.
- The expectation of this quantity is

$$\begin{aligned} E\Big\{\frac{1}{n}\sum_{i=1}^{n}\hat{f}_{h,-i}(X_i)\Big\} &= \frac{1}{n(n-1)}\sum_{i=1}^{n}\sum_{j\neq i}E\Big\{\frac{1}{h}K\Big(\frac{X_i - X_j}{h}\Big)\Big\} \\ &= E\Big\{\frac{1}{h}K\Big(\frac{X_1 - X_2}{h}\Big)\Big\} \end{aligned}$$

- The leave-one-out **cross-validation estimate of the MISE** (ignoring the irrelevant $\int f^2(x)dx$) is

$$\hat{J}_{MISE}(h) = \int \hat{f}_h^2(x)dx - \frac{2}{n}\sum_{i=1}^{n}\hat{f}_{h,-i}(X_i)$$

- The integrated squared error cross-validation choice of the bandwidth $h_{n,cv}$ is the value of $h > 0$ which minimizes $\hat{J}_{MISE}(h)$

$$h_{n,cv} = \arg\min_{h>0}\hat{J}_{MISE}(h)$$

8.5.2 Computing the Cross-validation Bandwidth

- **R** does have built-in capabilities for finding the bandwidth through cross-validation, but let's do an example ourselves to see how the process works.
- First, we can build a function called `J_mise` that has input and output:
  - Input: a value of the bandwidth h and a vector of data
  - Output: the value of $\hat{J}_{MISE}(h)$

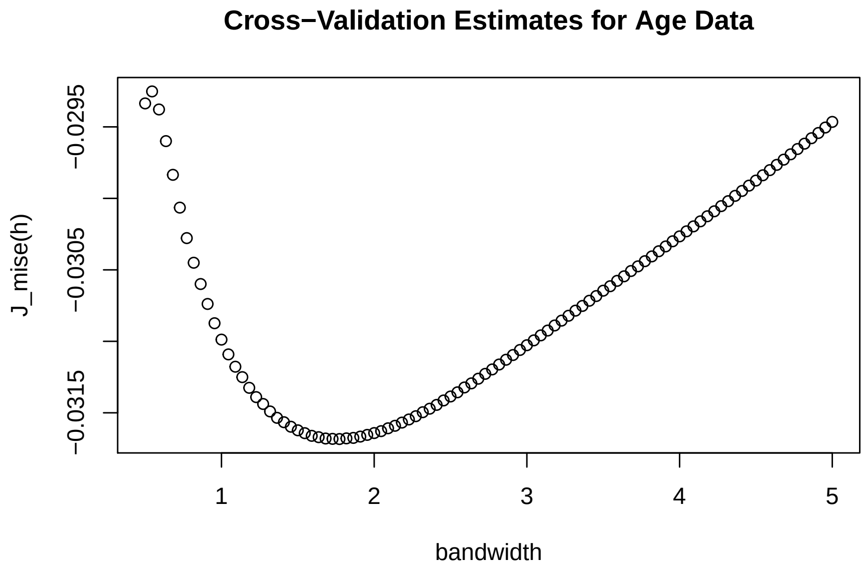
```r
J_mise <- function(h, x) {
    n <- length(x)
    loo.val <- rep(0, n)
    for(k in 1:n) {
        ## compute the leave-one-out density estimate
        ## only focus on density on interval (18, 90)
        loo.fhat <- density(x[-k], bw=h, from=18, to=90)
        loo.fhat.fn <- approxfun(loo.fhat$x, loo.fhat$y)
        loo.val[k] <- loo.fhat.fn(x[k])
    }
    fhat <- density(x, bw=h, from=18, to=90)
    fhat.sq.fn <- approxfun(fhat$x, fhat$y^2)

    ans <- integrate(fhat.sq.fn, lower=18, upper=90)$value - 2*mean(loo.val)
    return(ans)
}
```

- Now, we want to use our **R** function to compute $\hat{J}(h_j)$ over a grid of bandwidth values
$$h_{min} \leq h_1 < ... < h_q \leq h_{max}$$
- The smallest and largest possible bandwidths h_{min} and h_{max} can be chosen by looking at plots of the density for different bandwidths.
- h_{min} should correspond to a density estimate which is very nonsmooth while h_{max} should correspond to a density which is oversmoothed.
- For the ages from the kidney data, $h_{min} = 1/2$ and $h_{max} = 5$ seem like reasonable choices.
- Let us compute $\hat{J}(h)$ for a grid of 100 bandwidths between 1/2 and 5:

```r
h.grid <- seq(1/2, 5, length.out=100)
CV.est <- rep(0, 100)
for(j in 1:100) {
   CV.est[j] <- J_mise(h=h.grid[j], x=kidney$age)
}
```

- Now, if we plot $\hat{J}(h)$ vs. h, we can see what the best value of the bandwidth is. From the graph, it appears that a bandwidth of roughly $h = 1.8$ minimizes $\hat{J}(h)$

```r
plot(h.grid, CV.est, xlab="bandwidth", ylab="J_mise(h)",
     main = "Cross-Validation Estimates for Age Data")
```



- The specific value of the bandwidth where $\hat{J}(h_j)$ reaches its minimum is:

```r
h.grid[ which.min(CV.est) ]
```

```
## [1] 1.772727
```

8.5.3 Likelihood Cross-Validation

- An alternative to MISE is the **Kullback-Leibler (KL) divergence**

$$
\begin{aligned}
\mathrm{KL}(h) &= \int \log\Big(\frac{f(x)}{\hat{f}_h(x)}\Big) f(x)dx \\
&= -\int \log\{\hat{f}_h(x)\} f(x)dx + \int \log\{f(x)\} f(x)dx
\end{aligned}
$$

- We only need to get an estimate of $\int \log\{\hat{f}_h(x)\} f(x)dx$ because $\int \log\{f(x)\}f(x)dx$ does not depend on h.
- We can use the same approach as we did for integrated squared error cross-validation to estimate $\int \log\{\hat{f}_h(x)\}f(x)dx$.
- The leave-one-out **cross-validation estimate of the KL divergence** (ignoring the irrelevant $\int \log\{f(x)\}f(x)dx$ term) is

$$\hat{J}_{KL}(h) = -\frac{1}{n}\sum_{i=1}^{n} \log\{\hat{f}_{h,-i}(X_i)\}$$

- Choosing the bandwidth which minimizes $\hat{J}_{KL}(h)$ is often referred to as **likelihood cross-validation**.
- An **R** function which can compute $\hat{J}_{KL}(h)$ is given below:

```
J_KL <- function(h, x) {
  n <- length(x)
  loo.val <- rep(0, n)
  for(k in 1:n) {
    ## compute the leave-one-out density estimate
    ## only focus on density on interval (18, 90)
    loo.fhat <- density(x[-k], bw=h, from=18, to=90)
    loo.fhat.fn <- approxfun(loo.fhat$x, loo.fhat$y)
    loo.val[k] <- (-1)*log(loo.fhat.fn(x[k]))
  }
  return(mean(loo.val))
}
```

- **Exercise 8.7** Using the **age** variable from the kidney function data, find the best bandwidth using a Gaussian kernel and likelihood cross-validation.

8.6 Density Estimation in R

- In **R**, kernel density estimates are computed using the `density` function

```
density(x, bw, kernel, n, ...)
```

- **x** - the vector containing the data
- **bw** - the value of the bandwidth.
  - `bw = nrd0` gives the default bandwidth rule. This is Silverman's rule-of-thumb $h_n = 0.9sn^{-1/5}$
  - `bw = nrd` gives the bandwidth $h_n = 1.06\hat{\sigma}n^{-1/5}$
  - `bw = ucv` or `bw = bcv` find the bandwidth using cross-validation
- **kernel** - the choice of kernel function. The default kernel is the Gaussian kernel.

- Be careful, some of the non-Gaussian kernels used in the `density` function are scaled differently than the definitions you might often see in textbooks or on-line resources.
- **n** - the number of equally spaced points at which the density is to be estimated. The default is 512

- In this section, we will use the `galaxies` dataset in the `MASS` package. The first few observations of the `galaxies` dataset look like:

```
library(MASS)
galaxies[1:5]
```

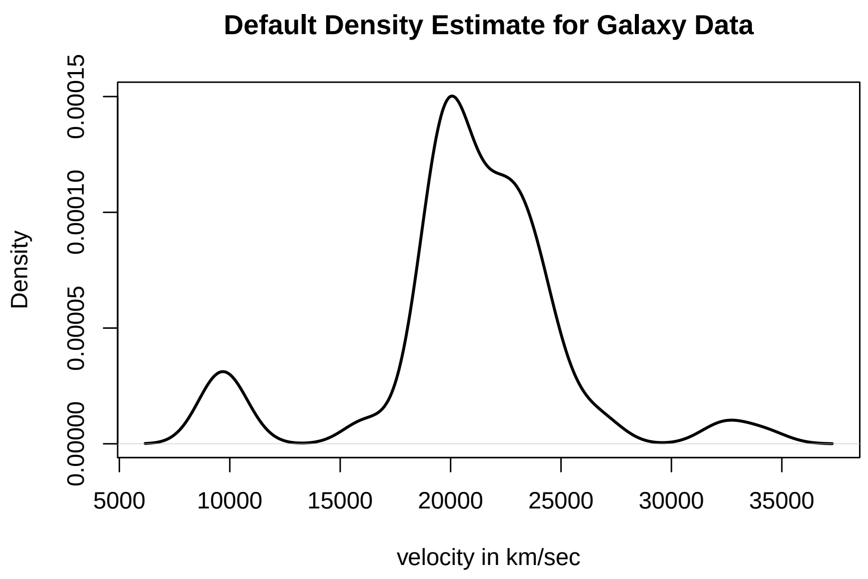
```
## [1] 9172 9350 9483 9558 9775
```

- Kernel density estimates can be computed in **R** using the `density` function

```
galax.dens <- density(galaxies)
```

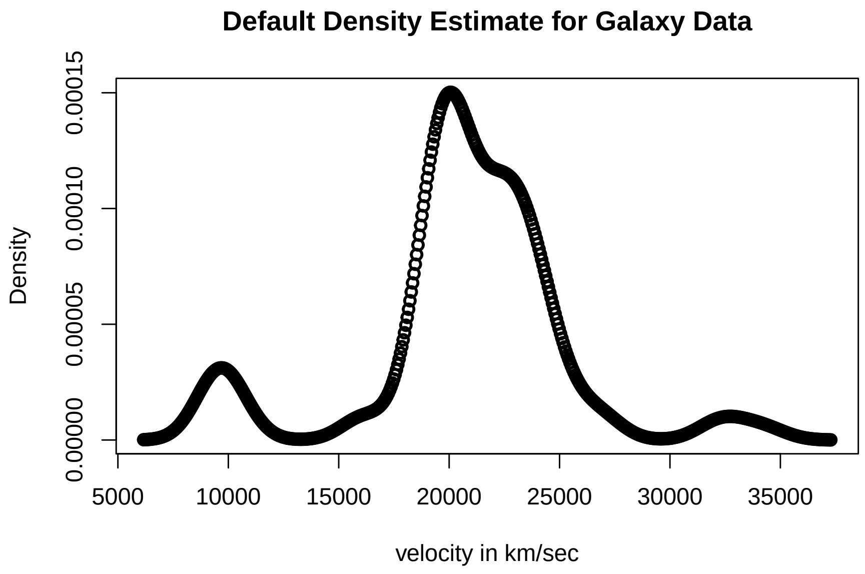
- You can display a density plot just by applying the `plot` function to our `galax.dens` object

```
plot(galax.dens, main="Default Density Estimate for Galaxy Data",
     xlab="velocity in km/sec", ylab="Density", lwd=2)
```



- The density function returns vectors `x` and `y` as components
  - `x` - the vector of points at which the density was estimated
  - `y` - the value of the estimated density at each of the `x` points
- So, just plotting the `(x, y)` should give you a plot of the density estimate

```
plot(galax.dens$x, galax.dens$y, main="Default Density Estimate for Galaxy Data",
     xlab="velocity in km/sec", ylab="Density", lwd=2)
```



- The default in **R** is to estimate the density at 512 points. Thus, `galax.dens$x` and `galax.dens$y` should each have length 512.

- The `bw` component returned by the `density` function is the bandwidth used to estimate the density.

```
galax.dens$bw
```

```
## [1] 1001.839
```

- The default bandwidth selection rule in **R** is Silverman's rule-of-thumb $h_n^{SR} = 0.9sn^{-1/5}$.
- We can check that this is true with the following code:

```
0.9*min(sd(galaxies), IQR(galaxies)/1.34)/(length(galaxies)^(1/5))
```

```
## [1] 1001.839
```

- The **R** function `approxfun` is useful if you want to compute the (approximate) value of the density estimate at a particular point. This approximation will be very close to the true value since the `density` function returns the value of the density over a dense grid of points.
- `approxfun` just linearly interpolates between a set of x and y values.
- Note that, as a default, `approxfun` returns NA values if you try to evaluate the function at points which are either less than the minimum x value or greater than maximum x value.
- For example, suppose we want to know the value of the density estimate at the points 18000 and 33000. This could be done with the following code

```
galaxy.fn <- approxfun(galax.dens$x, galax.dens$y)

galaxy.fn(18000)
```

```
## [1] 4.73857e-05
```

```
galaxy.fn(33000)
```

```
## [1] 1.004542e-05
```

- **Exercise 8.8** Plot the density estimate for the galaxy dataset using the following three rules for finding the bandwidth: (1) `bw = nrd`, (2) `bw=nrd0`, (3) `bw=ucv`. Based on these plots, how many distinct "clusters" or "groups" would you say the galaxy observations fall into.

- **Exercise 8.9** Consider a density estimate of the form

$$\tilde{f}(x) = \sum_{k=1}^{4} \pi_k \frac{1}{\sigma_k} K\Big(\frac{x - \mu_k}{\sigma_k}\Big),$$

  where $\sigma_k > 0$ and the π_k represent probabilities, that is, $\pi_k \geq 0$ with $\pi_1 + \pi_2 + \pi_3 + \pi_4 = 1$. Can you guess values for π_k, μ_k, and σ_k that fit the galaxy data well? Try finding the best values of (π_k, μ_k, σ_k) by plotting $\tilde{f}(x)$ next to the kernel density estimate returned by the `density` function in `R`.

8.7 Additional Reading

- Additional reading which covers the material discussed in this chapter includes:
  - Chapters 2-3 from Silverman (2018)
  - Chapter 6 from Wasserman (2006)
  - Chapters 2-3 from Härdle et al. (2012)
  - Chapter 4 from Izenman (2008)
  - Sheather (2004)

Part III

Quantifying Uncertainty

Chapter 9

The Bootstrap

9.1 Introduction

- The jackknife and the bootstrap are nonparametric procedures that are mainly used for finding standard errors and constructing confidence intervals.

Why use the bootstrap?

1. To find better standard errors and/or confidence intervals when the standard approximations do not work very well.
2. To find standard errors and/or confidence intervals when you have no idea how to compute reasonable standard errors.

Example: Inference for e^{μ}

- Suppose we have i.i.d. data $X_1, \ldots, X_n \sim \text{Logistic}(\mu, s)$, meaning that $E(X_i) = \mu$ and $\text{Var}(X_i) = \sigma^2 = s^2\pi^2/3$.

- Suppose our goal is to construct a confidence interval for the parameter $\theta = e^{\mu}$.

- The traditional approach to constructing a confidence interval uses the fact that
$$\sqrt{n}\Big(e^{\bar{X}} - e^{\mu}\Big) \longrightarrow \text{Normal}(0, \sigma^2 e^{2\mu})$$
so that we can assume $e^{\bar{X}}$ has a roughly Normal distribution with mean e^{μ} and standard deviation $\sigma e^{\mu}/\sqrt{n}$. This approximation is based on a Central Limit Theorem and "delta method" argument.

- The estimated standard error in this case is

$$\frac{\hat{\sigma} e^{\bar{X}}}{\sqrt{n}}$$

- Using this Normal approximation for $e^{\bar{X}}$, the 95% confidence interval for e^{μ} is

$$\Big[e^{\bar{X}} - 1.96 \times \frac{\hat{\sigma} e^{\bar{X}}}{\sqrt{n}}, e^{\bar{X}} + 1.96 \times \frac{\hat{\sigma} e^{\bar{X}}}{\sqrt{n}}\Big] \tag{9.1}$$

- For specific choices of n, how good is the Normal approximation for the distribution of $e^{\bar{X}}$?
- The figure below shows a histogram for the simulated distribution of $e^{\bar{X}}$ when $n = 50$. The density of the Normal approximation is also shown in this figure.

- As we can see from the above histogram, the Normal approximation is not terrible. However, it really does not capture the skewness in the distribution of $e^{\bar{X}}$ correctly.
- This could effect the coverage performance of confidence intervals which use Normal approximation (9.1).
- The bootstrap offers an alternative approach for constructing confidence intervals which does not depend on parametric approximations such as (9.1).

Example: Inference for the Correlation

- The sample correlation $\hat{\rho}$ which estimates the correlation $\rho = \text{Corr}(X_i, Y_i)$ between X_i and Y_i is defined as

$$\hat{\rho} = \frac{\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum_{i=1}^{n}(X_i - \bar{X})^2}\sqrt{\sum_{i=1}^{n}(Y_i - \bar{Y})}}$$

- Even such a relatively straightforward estimate has a pretty complicated formula for the estimated standard error if you use a multivariate delta method argument:

$$\text{std.err}_{corr} = \left\{\frac{\hat{\rho}^2}{4n}\left[\frac{\hat{\mu}_{40}}{\hat{\mu}_{20}^2} + \frac{\hat{\mu}_{04}}{\hat{\mu}_{02}^2} + \frac{2\hat{\mu}_{22}}{\hat{\mu}_{20}\hat{\mu}_{02}} + \frac{4\hat{\mu}_{22}}{\hat{\mu}_{11}^2} - \frac{4\hat{\mu}_{31}}{\hat{\mu}_{11}\hat{\mu}_{20}} - \frac{4\hat{\mu}_{13}}{\hat{\mu}_{11}\hat{\mu}_{02}}\right]\right\}^{1/2} \tag{9.2}$$

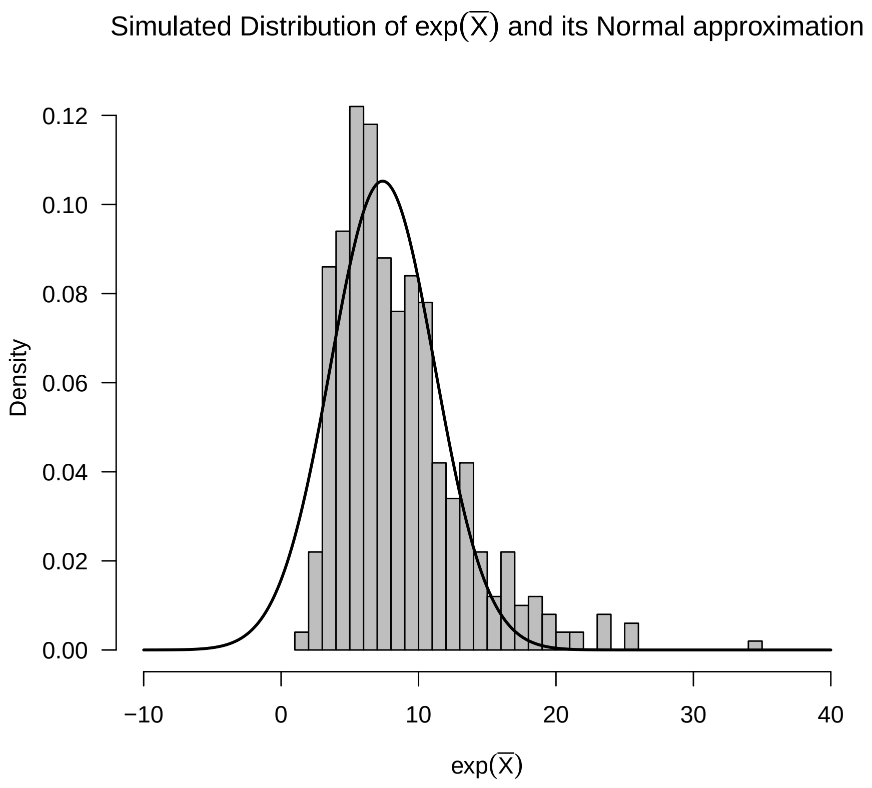


Figure 9.1: Histogram of simulated values of exp(sample mean) with density of the Normal approximation overlaid. This assumes n=50 and that the data are from a Logistic distribution with mu = 2 and s = 2.

where

$$\hat{\mu}_{hk} = \sum_{i=1}^{n} (X_i - \bar{X})^h (Y_i - \bar{Y})^k$$

- Another popular approach for constructing a confidence interval is to use Fisher's "z-transformation"

$$z = \frac{1}{2} \ln\Big(\frac{1+\hat{\rho}}{1-\hat{\rho}}\Big) \tag{9.3}$$

where it is argued that z has a roughly Normal distribution with mean $\frac{1}{2}\ln\{(1+\rho)/(1-\rho)\}$ and standard deviation $1/\sqrt{n-3}$.

- The bootstrap allows us to totally bypass the need to derive tedious formulas for the standard error such as (9.2) or bypass the need to use clever transformations such as (9.3).
- For many more complicated estimates deriving formulas such as (9.2) or transformations such as (9.3) may not even be feasible.
- The bootstrap provides an automatic way of constructing confidence intervals. You only need to be able to compute the estimate of interest.

9.2 Description of the Bootstrap

9.2.1 Description

- Suppose we have a statistic T_n that is an estimate of some quantity of interest θ.
- For example:
  - T_n - sample mean, $\theta = E(X_1)$.
  - T_n - sample correlation, $\theta = \text{Corr}(X_1, Y_1)$.
  - T_n - sample median, $\theta = F^{-1}(1/2)$; that is, the true median.
- Typically, T_n can be represented as a function of our sample $X_1, \dots, X_n$

$$T_n = h\Big(X_1, \dots, X_n\Big)$$

- Suppose we want to estimate the standard deviation of T_n.
- The true standard deviation of T_n is referred to as the **standard error**.

- Confidence intervals are often based on subtracting or adding an estimate of the standard error, e.g.

$$CI = T_n \pm z_{\alpha/2} \times \widehat{\text{standard}}\text{ error},$$

where $z_{\alpha/2}$ is the $100\times(1-\alpha/2)$ percentile of the Normal$(0,1)$ distribution.

- The bootstrap estimates the standard deviation of T_n by repeatedly subsampling from the original data and computing the value of the statistic T_n on each subsample.

- More generally, we can use the bootstrap not just to find the standard deviation of T_n but to characterize the distribution of T_n.

The Bootstrap Procedure

- In our description of the bootstrap, we will assume that we have the following ingredients:
  - $\mathbf{X} = (X_1, \ldots, X_n)$ where $X_1, \ldots, X_n$ are i.i.d. observations.
  - The statistic T_n of interest. $T_n = h(X_1, \ldots, X_n)$.
  - T_n is an estimate of θ.
  - R - the number of bootstrap replications.
- The bootstrap works in the following way:
- For $r = 1, \ldots, R$:
  - Draw a sample of size n: $(X_1^*, \ldots, X_n^*)$ by sampling with replacement from $\mathbf{X}$.
  - Compute $T_{n,r}^* = h(X_1^*, \ldots, X_n^*)$.

- Each sample is $(X_1^*, \ldots, X_n^*)$ is drawn through simple random sampling with replacement. That is, $X_1^*, \ldots, X_n^*$ are independent with

$$P(X_i^* = X_j) = \frac{1}{n} \quad \text{for } j = 1, \ldots, n$$

- We will refer to each sample $(X_1^*, \ldots, X_n^*)$ as a **bootstrap sample**.

- We will refer to $T_{n,r}^*$ as a **bootstrap replication**.

- The bootstrap estimate for the standard error of T_n is

$$se_{boot} = \left[\frac{1}{R-1} \sum_{r=1}^{R} \left(T_{n,r}^* - \frac{1}{R} \sum_{r=1}^{R} T_{n,r}^* \right)^2 \right]^{1/2}$$

- We can even use our bootstrap replications to get an approximation $\hat{G}_n^*(t)$ for the cumulative distribution function $G_n(t) = P(T_n \leq t)$ of T_n:

$$\hat{G}_n^*(t) = \frac{1}{R}\sum_{r=1}^{R} I\Big(T_{n,r}^* \leq t\Big)$$

- The **normal bootstrap standard error confidence interval** is defined as
$$\Big[T_n - z_{\alpha/2} se_{boot}, T_n + z_{\alpha/2} se_{boot}\Big]$$

- The **bootstrap percentile confidence interval** uses the percentiles of the boostrap replications $T_{n,1}^*, \ldots, T_{n,R}^*$ to form a confidence interval.

- The bootstrap $100\times\alpha/2$ and $100\times(1-\alpha/2)$ percentiles are roughly defined as

$$\begin{aligned} T_{[\alpha/2]}^{boot} &= \text{the point } t^* \text{ such that } 100\alpha/2 \text{ of the bootstrap replications are less than } t^* \\ T_{1-[\alpha/2]}^{boot} &= \text{the point } t^* \text{ such that } 100\alpha/2 \text{ of the bootstrap replications are less than } t^* \end{aligned}$$

- The level $100 \times (1-\alpha)\%$ level boostrap percentile confidence interval is then
$$\Big[T_{[\alpha/2]}^{boot}, T_{[1-\alpha/2]}^{boot}\Big]$$

- More precisely, the bootstrap percentiles are obtained by looking at the inverse of the estimated cdf of T_n

$$T_{[\alpha/2]}^{boot} = \hat{G}_n^{*,-1}(\alpha/2) \qquad T_{[1-\alpha/2]}^{boot} = \hat{G}_n^{*,-1}(1-\alpha/2)$$

- The bootstrap approach for computing estimated standard errors and confidence intervals is very appealing due to the fact that it is **automatic**.

- That is, we do not need to expend any effort deriving formulas for the variance of T_n and/or making asymptotic arguments for the distribution of T_n.

- We only need to be able to compute T_n many times, and the bootstrap procedure will automatically produce a confidence interval for us.

9.2.2 Example: Confidence Intervals for the Rate Parameter of an Exponential Distribution

- Suppose we have i.i.d. data $X_1, \dots, X_n$ from an Exponential distribution with rate parameter $1/\lambda$. That is, the pdf of X_i is

$$f(x) = \begin{cases} \frac{1}{\lambda}e^{-x/\lambda} & \text{if } x > 0 \\ 0 & \text{otherwise} \end{cases}$$

- This means that

$$E(X_i) = \lambda \quad \text{and} \quad \text{Var}(X_i) = \lambda^2$$

- If using the usual Normal approximation for constructing a confidence interval for λ, you would rely on the following asymptotic result:

$$\frac{\sqrt{n}(\bar{X} - \lambda)}{\bar{X}} \longrightarrow \text{Normal}(0, 1)$$

- In other words, for large n, $\bar{X}$ has an approximately Normal distribution with mean λ and standard deviation $\bar{X}/\sqrt{n}$.
- The estimated standard error in this case is $\bar{X}/\sqrt{n}$, and a 95% confidence interval for λ is

$$\left[\bar{X} - 1.96 \times \frac{\bar{X}}{\sqrt{n}}, \bar{X} + 1.96 \times \frac{\bar{X}}{\sqrt{n}}\right]$$

- Let's do a small simulation to see how the Normal approximation confidence interval compares with bootstrap-based confidence intervals.
- We will compare the Normal-approximation confidence interval with both the normal standard error bootstrap confidence interval and the percentile bootstrap confidence interval.

```r
xx <- rexp(50, rate=2) ## data, sample of 50 exponential r.v.s with mean 1/2
R <- 500   ## number of bootstrap replications
boot.mean <- rep(0, R)
for(r in 1:R) {
   boot.samp <- sample(1:50, size=50, replace=TRUE)
   xx.boot <- xx[boot.samp]   ## this is the bootstrap sample
   boot.mean[r] <- mean(xx.boot)  ## this is the rth bootstrap replication
}
```

```
par.ci <- c(mean(xx) - 1.96*mean(xx)/sqrt(50), mean(xx) + 1.96*mean(xx)/sqrt(50))
boot.ci.sd <- c(mean(xx) - 1.96*sd(boot.mean), mean(xx) + 1.96*sd(boot.mean))
boot.ci.quant <- quantile(boot.mean, probs=c(.025, .975))
```

- The normal-approximation confidence interval is

```
round(par.ci, 2)
```

```
## [1] 0.31 0.54
```

- The standard error boostrap confidence interval is

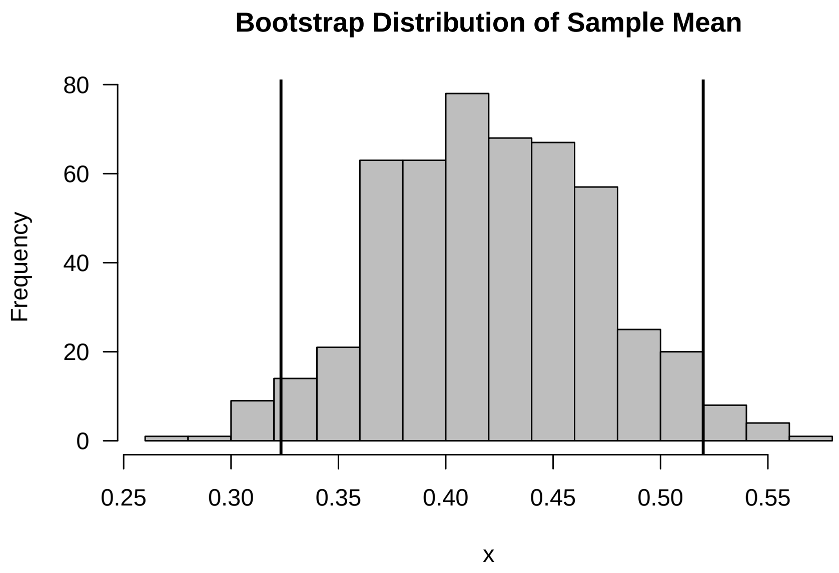
```
round(boot.ci.sd, 2)
```

```
## [1] 0.32 0.52
```

- The percentile bootstrap confidence interval

```
round(boot.ci.quant, 2)
```

```
##  2.5% 97.5%
##  0.32  0.52
```



9.2.3 Example: Confidence Intervals for the Ratio of Two Quantiles

- Suppose we have data from two groups

$$\begin{aligned}&\text{Group 1: } X_1, \dots, X_n \sim F_X \\ &\text{Group 2: } Y_1, \dots, Y_n \sim F_Y\end{aligned}$$

- The pth quantile for group 1 is defined as $\theta_{p1} = F_X^{-1}(p)$. In other words, if F_X is continuous then

$$P(X_i \leq \theta_{p1}) = F_X(F_X^{-1}(p)) = p$$

- Likewise, the pth quantile for group 2 is defined as $\theta_{p2} = F_Y^{-1}(p)$
- Suppose we are interested in estimating and constructing a confidence for the following parameter

$$\eta = \frac{\theta_{p1}}{\theta_{p2}}$$

- We will let $\hat{\theta}_{p1}$ denote the pth sample quantile from $(X_1, \dots, X_n)$ and let $\hat{\theta}_{p2}$ denote the pth sample quantile from $(Y_1, \dots, Y_n)$.
- We will estimate η with the ratio of the sample quantiles

$$\hat{\eta} = \frac{\hat{\theta}_{p1}}{\hat{\theta}_{p2}}$$

- It can be shown that

$$\hat{\theta}_{p1} \text{ has an approximate Normal}\Big(\theta_{p1}, \frac{p(1-p)}{nf_X^2(\theta_{p1})}\Big) \text{ distribution,}$$

  where $f_X(t) = F_X'(t)$ is the probability density function of X_i.
- Using a multivariate delta method argument, you can show that

$$\hat{\eta} \text{ has an approximate Normal}\Big(\eta, \frac{p(1-p)}{nf_X^2(\theta_{p1})\theta_{p2}^2} + \frac{p(1-p)\theta_{p1}^2}{nf_Y^2(\theta_{p2})\theta_{p2}^4}\Big) \text{ distribution} \tag{9.4}$$

- Using the above large-sample approximation, the estimated standard error that can be used to construct a confidence interval for η is

$$\sqrt{\frac{p(1-p)}{n\hat{f}_X^2(\hat{\theta}_{p1})\hat{\theta}_{p2}^2} + \frac{p(1-p)\hat{\theta}_{p1}^2}{n\hat{f}_Y^2(\hat{\theta}_{p2})\hat{\theta}_{p2}^4}}$$

- Let's do a small simulation study to see how the confidence interval based on the large-sample approximation (9.4) compares with bootstrap-based confidence intervals.
- We will simulate $X_i \sim \text{Gamma}(2, 1.5)$ and $Y_i \sim \text{Gamma}(2, 2)$ with $n = 100$ and $m = 100$.

```
n <- 100
m <- 100
xx <- rgamma(n, shape=2, rate=1.5)
yy <- rgamma(m, shape=2, rate=2)
```

- We will focus on estimating the pth quantile ratio for $p = 0.9$. In this case, the true value of η is $\eta \approx 4/3$.
- The estimate $\hat{\eta}$ and the estimated standard error using the large-sample approximation (9.4) is

```
theta.hat1 <- quantile(xx, probs=0.9)
theta.hat2 <- quantile(yy, probs=0.9)
eta.hat <- theta.hat1/theta.hat2    ## estimate of quantile ratio

xdensity <- density(xx)
ydensity <- density(yy)
fx <- approxfun(xdensity$x, xdensity$y)(theta.hat1)
fy <- approxfun(ydensity$x, ydensity$y)(theta.hat2)

q1.se.sq <- (.9*.1)/(n*(fx*theta.hat2)^2)
q2.se.sq <- (.9*.1*theta.hat1*theta.hat1)/(n*fy*fy*((theta.hat2)^4))
std.err <- sqrt(q1.se.sq + q2.se.sq)
```

- The confidence interval using the large-sample approximation (9.4) is

```
CI <- c(eta.hat - 1.96*std.err, eta.hat + 1.96*std.err)
round(CI, 2)
```

```
##  90%  90%
## 0.94 1.57
```

- Now, using the same simulated data, let's compute 500 bootstrap replications of the statistic $\hat{\eta}$

```
R <- 500
eta.boot <- numeric(R)

for(r in 1:R)
{
  boot.xx <- sample(xx, size=n, replace = TRUE)
  boot.yy <- sample(yy, size=m, replace = TRUE)
  thetahat.p1 <- quantile(boot.xx, probs=0.9)
  thetahat.p2 <- quantile(boot.yy, probs=0.9)
  eta.boot[r] <- thetahat.p1/thetahat.p2
}
```

- Because this is a two-sample setting, we draw bootstrap samples $(X_1^*, \ldots, X_n^*)$ and $(Y_1^*, \ldots, Y_m^*)$ for each group separately to generate each bootstrap replications.
- The standard error boostrap confidence interval is

```
boot.ci.sd <- c(eta.hat - 1.96*sd(eta.boot), eta.hat + 1.96*sd(eta.boot))

round(boot.ci.sd, 2)
```

```
## [1] 0.75 1.76
```

- The percentile bootstrap confidence interval is

```
boot.ci.quant <- quantile(eta.boot, probs=c(.025, .975))
round(boot.ci.quant, 2)
```

```
##  2.5% 97.5%
##  0.93  1.94
```

- A histogram of the bootstrap replications of $\hat{\eta}$ is shown in the below figure. Note that the true value of η is $\eta = 4/3$.

9.2.3.1 Comparing the Performance of the Bootstrap and Large-Sample Confidence Intervals

- We just saw that the bootstrap and the large-sample confidence intervals gave different answers.

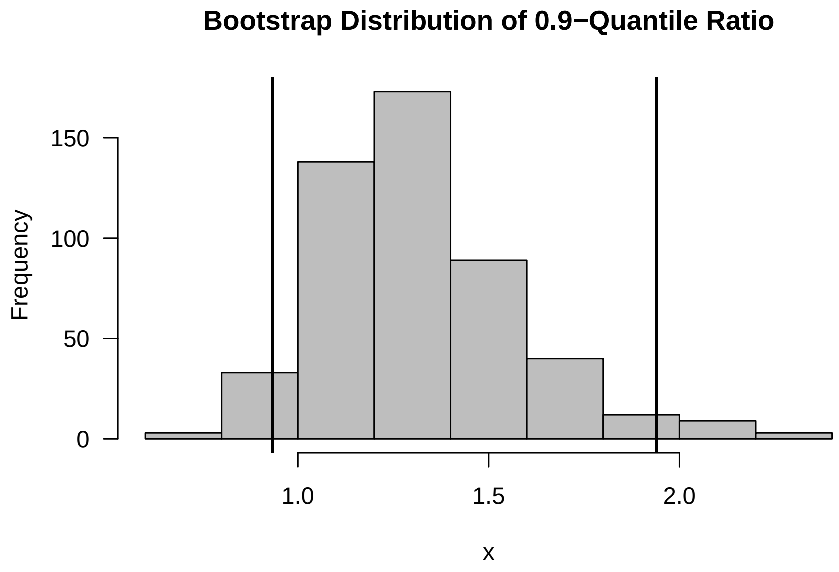


Figure 9.2: Bootstrap Distribution of the 0.9-Quantile Ratio. Vertical Lines are the Upper and Lower Bounds from the Percentile Bootstrap Confidence Interval.

- For this problem, what is the best approach for constructing confidence intervals?

- We can compare the performance of different confidence intervals by looking at their **coverage probability**.

- For a vector of data $\mathbf{X} = (X_1, \dots, X_n)$, we can represent a confidence interval for a parameter of interest θ as

$$\Big[L_{\alpha}(\mathbf{X}), U_{\alpha}(\mathbf{X})\Big] \tag{9.5}$$

  - $L_{\alpha}(\mathbf{X})$ is the lower confidence bound.
  - $U_{\alpha}(\mathbf{X})$ is the upper confidence bound.

- The coverage probability of a confidence interval $[L_{\alpha}(\mathbf{X}), U_{\alpha}(\mathbf{X})]$ is

$$P\Big(L_{\alpha}(\mathbf{X}) \leq \theta \leq U_{\alpha}(\mathbf{X})\Big),$$

  where we usually construct $L_{\alpha}(\mathbf{X})$ and $U_{\alpha}(\mathbf{X})$ so that the coverage probability is exactly equal or close to $1 - \alpha$.

- We can estimate this probability via simulation by looking at the following coverage proportion

$$\text{CoverProp}_{n_{rep}}(\theta) = \frac{1}{n_{rep}} \sum_{k=1}^{n_{rep}} I\Big(L_{\alpha}(\mathbf{X}^{(k)}) \leq \theta \leq U_{\alpha}(\mathbf{X}^{(k)}) \Big)$$

  - n_{rep} is the number of simulation replications
  - $X^{(k)}$ is the dataset from the k^{th} simulation replication
  - Each dataset $X^{(k)}$ is generated under the assumption that θ is the true value of the parameter of interest.

- We will compare coverage proportions using the same simulation design we used before for this quantile ratio example.
- That is, $p = 0.9$ and $X_i \sim \text{Gamma}(2, 1.5)$ and $Y_i \sim \text{Gamma}(2, 2)$ with $n = 100$ and $m = 100$.
- Below shows code for a simulation study which uses 1000 simulation replications. It compares the large-sample confidence interval which uses (9.4) with two bootstrap confidence intervals.

```r
n <- 100
m <- 100
R <- 500
eta.true <- 4/3

nreps <- 1000
Cover.par.ci <- numeric(nreps)
Cover.bootsd.ci <- numeric(nreps)
Cover.bootquant.ci <- numeric(nreps)
for(k in 1:nreps)  {
    ## Step 1: Generate the Data from Two Groups
    xx <- rgamma(n, shape=2, rate=1.5)
    yy <- rgamma(m, shape=2, rate=2)

    ## Step 2: Estimate eta from this data
    theta.hat1 <- quantile(xx, probs=0.9)
    theta.hat2 <- quantile(yy, probs=0.9)
    eta.hat <- theta.hat1/theta.hat2

    ## Step 3: Find confidence interval using large-sample Normal approximation
    xdensity <- density(xx)
    ydensity <- density(yy)
    fx <- approxfun(xdensity$x, xdensity$y)(theta.hat1)
    fy <- approxfun(ydensity$x, ydensity$y)(theta.hat2)
    q1.se.sq <- (.9*.1)/(n*(fx*theta.hat2)^2)
    q2.se.sq <- (.9*.1*theta.hat1*theta.hat1)/(n*fy*fy*((theta.hat2)^4))
    std.err <- sqrt(q1.se.sq + q2.se.sq)
    par.ci <- c(eta.hat - 1.96*std.err, eta.hat + 1.96*std.err)

    ## Step 4: Find bootstrap confidence intervals using R bootstrap replications
    eta.boot <- numeric(R)
    for(r in 1:R)   {
        boot.xx <- sample(xx, size=n, replace = TRUE)
        boot.yy <- sample(yy, size=m, replace = TRUE)
        thetahat.p1 <- quantile(boot.xx, probs=0.9)
        thetahat.p2 <- quantile(boot.yy, probs=0.9)
        eta.boot[r] <- thetahat.p1/thetahat.p2
    }
    boot.ci.sd <- c(eta.hat - 1.96*sd(eta.boot), eta.hat + 1.96*sd(eta.boot))
    boot.ci.quant <- quantile(eta.boot, probs=c(.025, .975))

    ## Step 5: Record if the true parameter is covered or not:
    Cover.par.ci[k] <- ifelse(par.ci[1] < eta.true & par.ci[2] >= eta.true, 1, 0)
    Cover.bootsd.ci[k] <- ifelse(boot.ci.sd[1] < eta.true & boot.ci.sd[2] >= eta.true,
                                 1, 0)
    Cover.bootquant.ci[k] <- ifelse(boot.ci.quant[1] < eta.true &
                                        boot.ci.quant[2] >= eta.true, 1, 0)
}
```

- The coverage proportions for each of the methods are:

```
mean(Cover.par.ci)
```

```
## [1] 0.921
```

```
mean(Cover.bootsd.ci)
```

```
## [1] 0.949
```

```
mean(Cover.bootquant.ci)
```

```
## [1] 0.959
```

- Using these simulated outcomes, we can also construct 95% confidence intervals for the coverage probabilities:

	Lower CI	Estimate	Upper CI
Large-Sample Approximation	0.904	0.921	0.938
Bootstrap Std. Err.	0.935	0.949	0.963
Bootstrap Percentile	0.947	0.959	0.971

Table 9.1: Estimates and Confidence Intervals for the Coverage Probabilities of Different Methods.

9.3 Why is the Bootstrap Procedure Reasonable?

- As mentioned before, our original motivation for the bootstrap was to find an estimate of $\text{Var}(T_n)$ where T_n is a statistic that can be thought of as an estimate of θ.
- The statistic T_n can be thought of as a function of our sample

$$T_n = h(X_1, \dots, X_n)$$

  where $X_1, \dots, X_n$ is an i.i.d. sample with cumulative distribution function F.
- If we had a way of simulating data $X_i^{(k)}$ from F, we could estimate $\text{Var}(T_n)$ with the following quantity

$$\begin{aligned}\widehat{\text{Var}(T_n)} &= \frac{1}{K-1}\sum_{k=1}^{K}\Big(h(X_1^{(k)}, \dots, X_n^{(k)}) - \frac{1}{K}\sum_{k=1}^{K} h(X_1^{(k)}, \dots, X_n^{(k)})\Big)^2 \\ &= \frac{1}{K-1}\sum_{k=1}^{K}\Big(T_n^{(k)} - \frac{1}{K}\sum_{k=1}^{K} T_n^{(k)}\Big)^2 \qquad (9.6)\end{aligned}$$

  - $X_1^{(k)}, \dots, X_n^{(k)}$ is an i.i.d. sample from F.
  - $T_n^{(k)} = h(X_1^{(k)}, \dots, X_n^{(k)})$ is the value of out statistic of interest from the k^{th} simulated dataset.

- In practice, F is unknown, and we cannot simulate data from F.
- The main idea behind the bootstrap is that the empirical distribution function $\hat{F}_n$ is a very good estimate of F.
- Hence, if we sample $X_1^{(k)}, \dots, X_n^{(k)}$ from $\hat{F}_n$ instead of F and use the formula (9.6), this should give us a good estimate of the variance of T_n.

- How can we simulate data from $\hat{F}_n$?
- Recall that $\hat{F}_n$ is a discrete distribution that has mass $1/n$ at each of the observed data points $\mathbf{X} = (X_1, \dots, X_n)$.
- So, if we say $X_i^* \sim \hat{F}_n$, then

$$P(X_i^* = X_j) = 1/n \qquad \text{for } j = 1, \dots, n, \tag{9.7}$$

  where $(X_1, \dots, X_n)$ can be thought of as fixed numbers.
- To simulate a random variable $X_i^* \sim \hat{F}_n$, we just need to draw one of the observations X_j from $\mathbf{X}$ at random and set $X_i^* = X_j$.
- Then, to simulate an i.i.d. sample $X_1^*, \dots, X_n^*$ from $\hat{F}_n$, we just to sample the X_i^* from $\mathbf{X}$ with replacement. In other words, we just need to use the same procedure we discussed earlier for generating bootstrap samples.
- Each bootstrap sample $(X_1^*, \dots, X_n^*)$ can be thought of as an i.i.d. sample from $\hat{F}_n$.

- The variance of T_n can be written as

$$V_{T_n}(F) = \int \cdots \int h^2(x_1, \dots, x_n) dF(x_1) \cdots dF(x_n) - E_{T_n}^2(F),$$

  where

$$E_{T_n}(F) = \int \cdots \int h(x_1, \dots, x_n) dF(x_1) \cdots dF(x_n),$$

- What we are trying to compute with the bootstrap is the following variance estimate

$$V_{T_n}(\hat{F}_n) = \int \cdots \int h^2(x_1, \dots, x_n) d\hat{F}_n(x_1) \cdots d\hat{F}_n(x_n) - E_{T_n}^2(\hat{F}_n) \tag{9.8}$$

- It is too difficult to compute (9.8) in most cases. Instead, with the bootstrap, we are approximating (9.8) via simulation by drawing many i.i.d. samples from $\hat{F}_n$.
- You can think of the bootstrap as using Monte Carlo integration to approximate (9.8).

- There are cases where the bootstrap does not really work well.

- The main requirement for the bootstrap to work is that the functional $E_{T_n}(F)$ is "smooth" as F varies.
- That is, if F_1 and F_2 are "close", then $E_{T_n}(F_1)$ and $E_{T_n}(F_2)$ should also be close.
- More specifically, if the functional $E_{T_n}(F)$ is differentiable in an appropriate sense, then confidence intervals from the bootstrap estimate of the variance of T_n will be "valid" in an asymptotic sense (see, for example, Chapter 5 of Shao (2003) for a somewhat more rigorous discussion of this).

9.4 Pivotal Bootstrap Confidence Intervals

- Confidence intervals are often based on what is referred to as a **pivot**.
- The quantity $W_n(\mathbf{X}, \theta)$ is a pivot if the distribution of $W_n(\mathbf{X}, \theta)$ does not depend on θ.
- A common example of this is for the Normal(θ, σ^2) distribution. In this case,
$$W_n(\mathbf{X}, \theta) = \bar{X} - \theta \tag{9.9}$$
is a pivot. The distribution of $W_n(\mathbf{X}, \theta)$ is Normal$(0, \sigma^2)$.
- Using the pivot allows us to construct a confidence interval because
$$\begin{aligned} 1 - \alpha &= P\Big(-\sigma z_{1-\alpha/2} \leq W_n(\mathbf{X}, \theta) \leq \sigma z_{1-\alpha/2} \Big) \\ &= P\Big(-\sigma z_{1-\alpha/2} \leq \bar{X} - \theta \leq \sigma z_{1-\alpha/2} \Big) \\ &= P\Big(\bar{X} - \sigma z_{1-\alpha/2} \leq \theta \leq \bar{X} + \sigma z_{1-\alpha/2} \Big) \end{aligned}$$
- Another common pivot (if we did not assume σ^2 was known) would be
$$W_n(\mathbf{X}, \theta) = \frac{\sqrt{n}(\bar{X} - \theta)}{\hat{\sigma}},$$
which would have a t distribution.

- We can use a similar idea to construct a bootstrap confidence interval for $\theta = E(T_n)$.
- Assume that $W_n(\mathbf{X}, \theta) = T_n - \theta$ is a pivot and suppose that $H(t)$ is the cdf of this pivot

- Then, if we choose $b > a$ such that $H(b) - H(a) = 1 - \alpha$,

$$\begin{aligned} 1-\alpha &= P\Big(a \leq T_n - \theta \leq b) = P\Big(-b \leq \theta - T_n \leq -a\Big) \\ &= P\Big(T_n - b \leq \theta \leq T_n - a\Big) \end{aligned}$$

  For example, $b = H^{-1}(1-\alpha/2)$ and $a = H^{-1}(\alpha/2)$ would work.

- This would suggest that $[T_n - b, T_n - a]$ should be a good confidence interval for θ.
- The only problem is that $H(t)$ is not known. So, how do we find a and b if we don't assume normality of T_n and a known σ^2?
- **Idea:** Look at the distribution of $T^*_{n,r} - T_n$ as a substitute for $T_n - \theta$ and use the empirical distribution function of $T^*_{n,r} - T_n$ to estimate $H(t)$

$$\hat{H}_R(t) = \frac{1}{R}\sum_{r=1}^{R} I\Big(T^*_{n,r} - T_n \leq t\Big) \tag{9.10}$$

- Using this approximation for $H(t)$, we could use the following confidence interval

$$\Big[T_n - \hat{H}_R^{-1}(1-\alpha/2), T_n - \hat{H}_R^{-1}(\alpha/2)\Big]$$

Studentized Bootstrap Confidence Intervals

- Now, suppose we instead use the pivot

$$\mathbf{Z}_n(\mathbf{X}, \theta) = \frac{T_n - \theta}{se_{boot}},$$

  where se_{boot} denotes the bootstrap estimate of standard error.

- We will let $K(t)$ denote the cdf of $\mathbf{Z}_n(\mathbf{X}, \theta)$.
- Using the same reasoning as before,

$$\begin{aligned} 1-\alpha &= P\Big(K^{-1}(\alpha/2) \leq \frac{T_n - \theta}{se_{boot}} \leq K^{-1}(1-\alpha/2)\Big) \\ &= P\Big(- se_{boot} \times K^{-1}(1-\alpha/2) \leq \theta - T_n \leq -se_{boot} \times K^{-1}(\alpha/2)\Big) \\ &= P\Big(T_n - se_{boot} \times K^{-1}(1-\alpha/2) \leq \theta \leq T_n - se_{boot} \times K^{-1}(\alpha/2)\Big) \end{aligned}$$

  and hence a confidence interval for θ would be

$$\Big[T_n - se_{boot} \times K^{-1}(1-\alpha/2), T_n - se_{boot} \times K^{-1}(\alpha/2)\Big]$$

- To estimate $K(t)$, we are going to use $Z^*_{n,r} = (T^*_{n,r} - T_n)/\hat{se}_r$ as a substitute for $(T_n - \theta)/se_{boot}$.
- The estimate $\hat{se}_r$ is an estimate of the standard error of $T^*_{n,r}$. This could be estimated via

$$\hat{se}_r = \left[\frac{1}{J-1}\sum_{j=1}^{J}\Big(T^{**}_{n,r,j} - \frac{1}{J}\sum_{j=1}^{J} T^{**}_{n,r,j}\Big)^2\right]^{1/2},$$

  where $T^{**}_{n,r,j}$ is the value of our test statistic computed from the j^{th} bootstrap sample of the bootstrap sample that was used to produce $T^*_{n,r}$.
- So, to find $\hat{se}_r$, we need J bootstrap samples within each of the R bootstrap samples that were used to generated $T^*_{n,1}, \ldots, T^*_{n,R}$. For this reason, this is often referred to as the double bootstrap.
- Then, the estimate of $K(t)$ is defined as

$$\hat{K}_R(t) = \frac{1}{R}\sum_{r=1}^{R} I\Big(Z^*_{n,r} \leq t\Big)$$

- The **studentized** bootstrap confidence interval (or the **bootstrap-t** confidence interval) is then defined as

$$\Big[T_n - se_{boot} \times \hat{K}_R^{-1}(1-\alpha/2), T_n - se_{boot} \times \hat{K}_R^{-1}(\alpha/2)\Big]$$

9.5 The Parametric Bootstrap

- With the bootstrap, we generate each bootstrap sample $(X^*_1, \ldots, X^*_n)$ by sampling with replacement from the empirical distribution function $\hat{F}_n$.
- For this reason, it can be referred to as the **nonparametric bootstrap**.
- With the **parametric bootstrap**, we sample from a parametric estimate $F_{\hat{\varphi}}$ of the cumulative distribution function instead of sampling from $\hat{F}_n$.
- For example, suppose we have data $(X_1, \ldots, X_n)$ that we assume are normally distributed and we estimate μ and σ^2 with

$$\hat{\mu} = \bar{X} \qquad \hat{\sigma}^2 = \frac{1}{n-1}\sum_{i=1}^{n}(X_i - \bar{X})^2 \tag{9.11}$$

- If we are interested in getting a confidence interval for θ where $T_n = h(X_1, \ldots, X_n)$ is an estimate of θ, the parametric bootstrap would use the following procedure:

- For $r = 1, \dots, R$:
  - Draw an i.i.d. sample: $X_1^*, \dots, X_n^* \sim \text{Normal}(\hat{\mu}, \hat{\sigma}^2)$.
  - Compute $T_{n,r}^* = h(X_1^*, \dots, X_n^*)$.
- Then, form a confidence interval for θ using the parametric bootstrap replications $T_{n,1}^*, \dots, T_{n,R}^*$

When Might the Parametric Bootstrap be Useful?

- In cases with smaller sample sizes. For small sample sizes, $F_{\hat{\varphi}}$ is often a better estimate than $\hat{F}_n$.
- When you are only interested in constructing a confidence interval for the parameter of a parametric model that you think fits the data well.
- For non-i.i.d. data or other complicated distributions, a parametric bootstrap can sometimes be easier to work with.

- For non-i.i.d. data, where our observations $(X_1, \dots, X_n)$ our dependent, a parametric bootstrap can often be straightforward to implement.
- Suppose G_{φ} is a parametric model that describes the joint distribution of $(X_1, \dots, X_n)$ and that it is easy to simulate observations from G_{φ}.
- Then, if you have an estimate $\hat{\varphi}$ of φ, you can use the following procedure to generate bootstrap replications for your statistic of interest.
- For $r = 1, \dots, R$:
  - Draw a sample $(X_1^*, \dots, X_n^*) \sim G_{\hat{\varphi}}$.
  - Compute $T_{n,r}^* = h(X_1^*, \dots, X_n^*)$.

9.5.1 Parametric Bootstrap for the Median Age from the Kidney Data

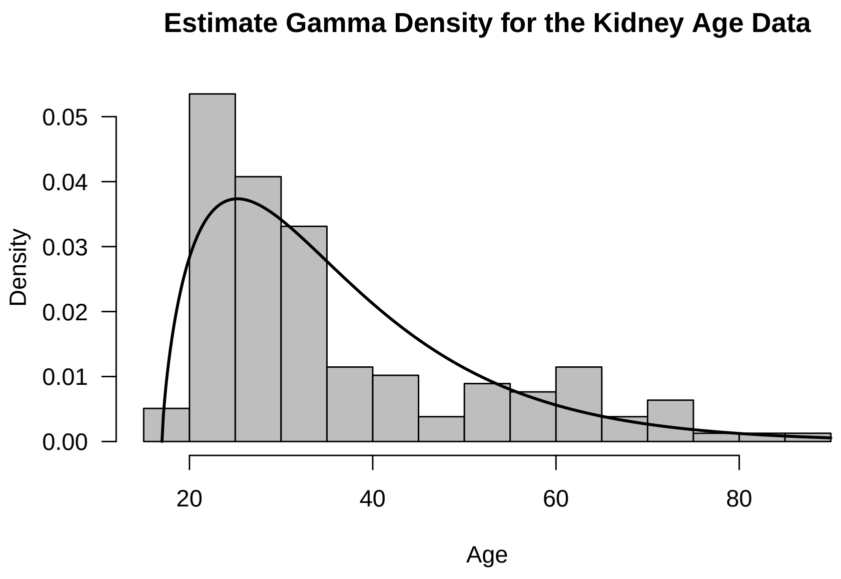
- Let us consider the ages from the kidney data.
- One somewhat reasonable model for the ages X_i from the kidney data is that
$$X_i - 17 \sim \text{Gamma}(\alpha, \beta)$$

- We can find the maximum likelihood estimates $(\hat{\alpha}, \hat{\beta})$ for this model using the following R code

```
kidney <- read.table("https://web.stanford.edu/~hastie/CASI_files/DATA/kidney.txt",
                     header=TRUE)

LogProfLik <- function(alpha, x) {
    ans <- alpha*log(alpha/mean(x)) - lgamma(alpha) + (alpha - 1)*mean(log(x)) - alpha
    return(ans)
}

best.alpha <- optimize(LogProfLik, interval=c(0,10), x=kidney$age - 17,
                       maximum=TRUE)$maximum
best.beta <- best.alpha/mean(kidney$age - 17)
```

- We can plot this estimated Gamma density overlaid on the histogram of the ages to see how they compare

```
tt <- seq(17, 90, length.out=500)
hist(kidney$age, breaks="FD", probability=TRUE, las=1, xlab="Age",
     main="Estimate Gamma Density for the Kidney Age Data", col="grey")
lines(tt, dgamma(tt - 17, shape=best.alpha, rate=best.beta), lwd=2)
```



- Suppose we are interested in constructing a confidence interval for the median age.
- To use the parametric boostrap with the parametric model $X_i - 17 \sim \text{Gamma}(\alpha, \beta)$ and using our estimates $\hat{\alpha}$ and $\hat{\beta}$ computed above, we would use the following steps.
- For $r = 1, \dots, R$:
  - Draw an i.i.d. sample: $X_1^*, \dots, X_n^* \sim 17 + \text{Gamma}(\hat{\alpha}, \hat{\beta})$.
  - Compute $T_{n,r}^* = \text{median}(X_1^*, \dots, X_n^*)$.
- The code for implementing this parametric bootstrap is given below

```
R <- 500
med.boot.par <- rep(0, R)
med.boot.np <- rep(0, R)
for(r in 1:R) {
    xx.boot.par <- 17 + rgamma(157, shape=best.alpha, rate=best.beta)
    xx.boot.np <- sample(kidney$age, size=157, replace=TRUE)

    med.boot.par[r] <- median(xx.boot.par)  ## rth par. bootstrap replication
    med.boot.np[r] <- median(xx.boot.np)  ## rth par. bootstrap replication
}
```

- The normal standard error confidence interval using the parametric bootstrap is

```
c(median(kidney$age) - 1.96*sd(med.boot.par), median(kidney$age) + 1.96*sd(med.boot.pa
```

```
## [1] 28.39858 33.60142
```

- The normal standard error confidence interval using the nonparametric bootstrap is

```
c(median(kidney$age) - 1.96*sd(med.boot.np), median(kidney$age) + 1.96*sd(med.boot.np)
```

```
## [1] 28.64083 33.35917
```

9.6 Additional Reading

- Additional reading which covers the material discussed in this chapter includes:
  - Chapter 3 from Wasserman (2006)
  - Chapter 2 from Davison and Hinkley (1997)

9.7 Exercises

Chapter 10

Bootstrap Examples and the Jackknife

10.1 The Parametric Bootstrap for an AR(1) model

- Consider the time series $X_1, X_2, \dots, X_m$. Here, X_t denotes an observation made at time t.
- An autoregressive model of order 1 (usually called an AR(1) model) for this time series is

$$\begin{aligned} X_1 &= \frac{c_0}{1-\alpha} + \varepsilon_1 \\ X_t &= c_0 + \alpha X_{t-1} + \varepsilon_t, \qquad t = 2, \dots, m. \end{aligned}$$

- It is usually assumed that $|\alpha| < 1$.
- In the AR(1) model, it is assumed that
  - $E(\varepsilon_t) = 0$
  - $\mathrm{Var}(\varepsilon_t) = \sigma^2$,
  - $\varepsilon_2, \dots, \varepsilon_m$ are i.i.d.
  - ε_t and X_{t-1} are independent.
- In addition to these assumptions, we will assume that

$$\varepsilon_t \sim \mathrm{Normal}(0, \sigma^2)$$

- The AR(1) model implies that

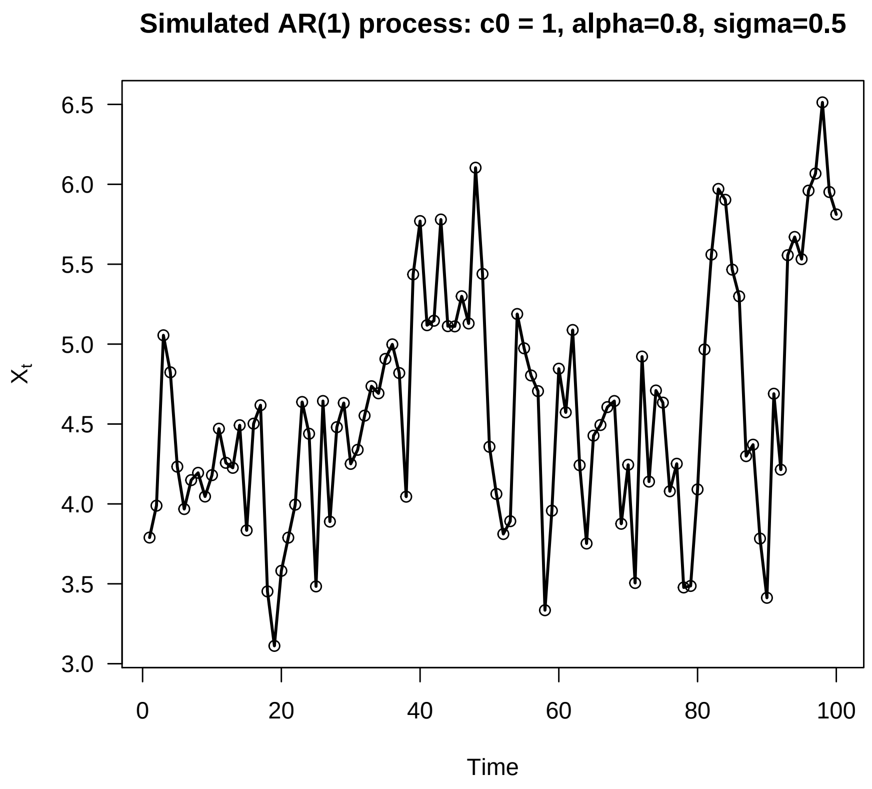
$$\mathrm{Corr}(X_t, X_{t-1}) = \alpha$$

and, more generally, that

$$\mathrm{Corr}(X_t, X_{t-p}) = \alpha^p$$

- For known values of c_0, α, and σ^2, we can simulate an AR(1) time series with the following `R` code:

```r
SimulateParAR1 <- function(m, c0, alpha, sig.sq) {
     xx <- numeric(m)
     xx[1] <- c0/(1 - alpha) + rnorm(1, sd=sqrt(sig.sq))
     for(t in 2:m) {
         xx[t] <- c0 + alpha*xx[t-1] + rnorm(1, sd=sqrt(sig.sq))
     }
     return(xx)
}
```



- In R, estimates of c_0, α, and σ^2 can be found by using the `ar` function. For example,

```
x <- SimulateParAR1(1000, 1, 0.8, sig.sq=.25)
ar1.fit <- ar(x, aic=FALSE, order.max = 1, method="mle")

c0.est <- ar1.fit$x.mean*(1 - ar1.fit$ar)
alpha.est <- ar1.fit$ar
sigsq.est <- ar1.fit$var.pred
```

- Suppose we want to construct confidence intervals for α and σ using a bootstrap method.

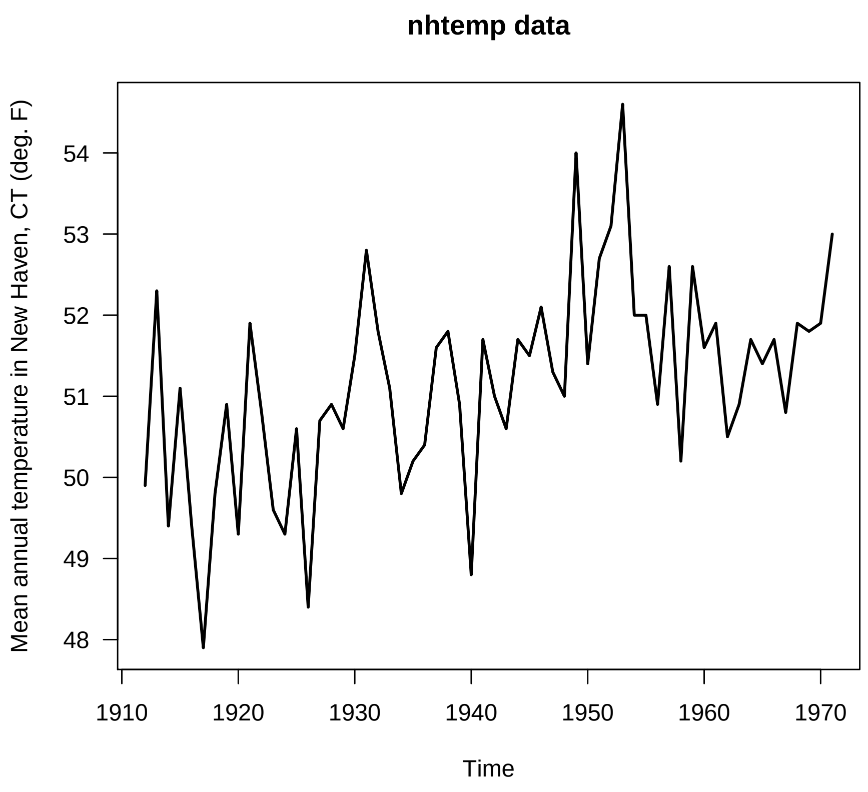
- Using the direct, nonparametric bootstrap described in the previous chapter will not work because our observations are not independent. There are "block bootstraps" that are designed to work for time series, but we will not discuss those here (see e.g., Bühlmann (2002) or Chapter 8 of Davison and Hinkley (1997) for more details).

- With the parametric bootstrap, we only have to use the following steps to generate bootstrap replications $\hat{\alpha}_r^*$ and $\hat{\sigma}_r^{2,*}$ for estimates of α and $\hat{\sigma}^2$.

- For $r = 1, \ldots, R$:

  - Simulate a time series $X_1^*, \ldots, X_m^*$ from an AR(1) model with parameters $(\hat{c}_0, \hat{\alpha}, \hat{\sigma}^2)$.
  - Compute $\hat{\alpha}_r^* = \hat{\alpha}(X_1^*, \ldots, X_m^*)$.
  - Compute $\hat{\sigma}_r^{2,*} = \hat{\sigma}^2(X_1^*, \ldots, X_m^*)$

- To see how this parametric bootstrap works, we will use the `nhtemp` dataset that is available in R.



- The `nhtemp` dataset contains the mean annual temperature in New Haven, Connecticut from the years 1912-1971

```
head(nhtemp)
```

```
## [1] 49.9 52.3 49.4 51.1 49.4 47.9
```

- The estimated autocorrelation parameter α is about 0.31 for this data

```
ar1.temp <- ar(nhtemp, aic=FALSE, order.max = 1)
c0.hat <- ar1.temp$x.mean*(1 - ar1.temp$ar)
alpha.hat <- ar1.temp$ar
sigsq.hat <- ar1.temp$var.pred
alpha.hat
```

```
## [1] 0.3148269
```

- Now, that we have estimated all the parameter of the AR(1) model, we can run our parametric bootstrap for $\hat{\alpha}$ and $\hat{\sigma}$:

```
R <- 500
alpha.boot <- numeric(R)
sigsq.boot <- numeric(R)
for(r in 1:R) {
  x <- SimulateParAR1(60, c0=c0.hat, alpha=alpha.hat, sig.sq=sigsq.hat)
  ar1.fit <- ar(x, aic=FALSE, order.max = 1)

  alpha.boot[r] <- ar1.fit$ar
  sigsq.boot[r] <- ar1.fit$var.pred
}
```

- Normal bootstrap standard error confidence intervals for α and σ^2 are

```
round(c(alpha.hat - 1.96*sd(alpha.boot), alpha.hat + 1.96*sd(alpha.boot)), 3)
```

```
## [1] 0.075 0.555
```

```
round(c(sigsq.hat - 1.96*sd(sigsq.boot), sigsq.hat + 1.96*sd(sigsq.boot)), 3)
```

```
## [1] 0.932 2.004
```

- We can compare our confidence interval for α with the confidence interval obtained from using a large-sample approximation:

```
asymp.se <- sqrt(ar1.temp$asy.var.coef)
round(c(alpha.hat - 1.96*asymp.se, alpha.hat + 1.96*asymp.se), 3)
```

```
## [1] 0.071 0.559
```

10.2 Using the Bootstrap in Regression

- In linear regression with a single, univariate covariate, we work with the following model

$$Y_i = \beta_0 + \beta_1 x_i + \varepsilon_i, \qquad i = 1, \dots, n.$$

  - Y_i - the responses
  - x_i - the covariates

  - β_0, β_1 - the regression coefficients
  - ε_i - the residuals

- Typically, confidence intervals for the regression coefficients β_0 and β_1 are constructed under the assumption that $\varepsilon_i \sim \text{Normal}(0, \sigma^2)$.
- The bootstrap allows us to compute confidence intervals for (β_0, β_1) without relying on this normality assumption.
- How to compute bootstrap confidence intervals for β_0 and β_1?

- The least-squares estimates of β_0 and β_1 are

$$\hat{\beta}_0 = \bar{y} - \hat{\beta}_1\bar{x} \qquad\qquad \hat{\beta}_1 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{S_{xx}}$$

where $S_{xx} = \sum_{i=1}^{n}(x_i - \bar{x})^2$.

- Assuming the covariates are fixed design points, the variance of $\hat{\beta}_0$ and $\hat{\beta}_1$ are

$$\text{Var}(\hat{\beta}_0) = \sigma^2\Big(\frac{\frac{1}{n}\sum_{i=1}^{n}x_i^2}{S_{xx}}\Big) \qquad \text{Var}(\hat{\beta}_1) = \frac{\sigma^2}{S_{xx}} \tag{10.1}$$

10.2.1 Parametric Bootstrap for Regression

- With a parametric bootstrap, we will simulate outcomes Y_i from the model

$$Y_i = \hat{\beta}_0 + \hat{\beta}_1 x_i + \varepsilon_i,$$

  - $\hat{\beta}_0$ and $\hat{\beta}_1$ are the least-squares estimates of β_0 and β_1,
  - $\varepsilon_1, \ldots, \varepsilon_n$ are i.i.d. random variables with mean zero and variance $\hat{\sigma}^2$.

- It is most common to assume that $\varepsilon_i \sim \text{Normal}(0, \hat{\sigma}^2)$, where $\hat{\sigma}^2$ is an estimate of the residual variance.
- However, we could easily use an alternative parametric model for ε_i if we thought it was appropriate.

- A t-distribution with a small number of degrees of freedom can be useful when the residuals are thought to follow a distribution with "heavier tails".
- If we assume $\varepsilon_i \sim \sigma \times t_3$, then $\text{Var}(\varepsilon_i) = 3\sigma^2$.

- So, with a t_3 residual distribution we want to simulate from the model

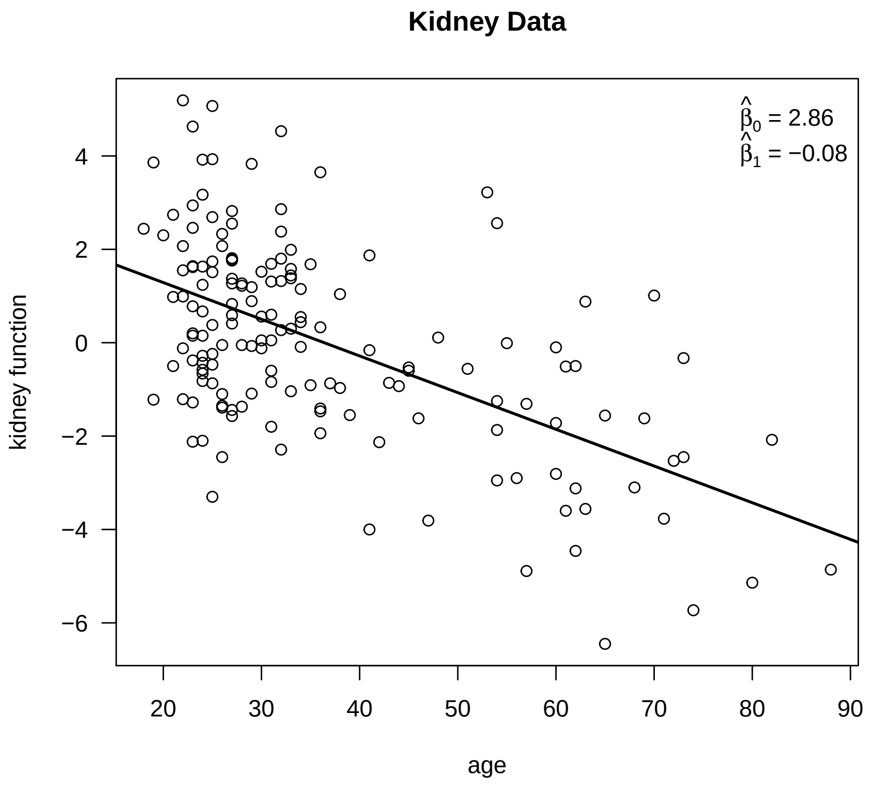
$$Y_i = \hat{\beta}_0 + \hat{\beta}_1 x_i + \frac{\hat{\sigma}}{\sqrt{3}} u_i, \qquad u_i \sim t_3,$$

where $\hat{\sigma}^2$ is the following estimate of the residual variance:

$$\hat{\sigma}^2 = \frac{1}{n-2}\sum_{i=1}^{n}(Y_i - \hat{\beta}_0 - \hat{\beta}_1)^2$$

- To show how this parametric-t bootstrap works in practice we will look at the kidney function data.
- We will look at a linear regression where the measure of kidney function is the outcome and age is the covariate.

```
kidney <- read.table("https://web.stanford.edu/~hastie/CASI_files/DATA/kidney.txt",
                     header=TRUE)
```



- Bootstrap replications of $\hat{\beta}_0$ and $\hat{\beta}_1$ can be computed using the following `R` code:

```r
## First find the parameter estimates
lm.kidney <- lm(tot ~ age, data=kidney)
beta0.hat <- lm.kidney$coef[1]
beta1.hat <- lm.kidney$coef[2]
sigsq.hat <- sum(lm.kidney$residuals^2)/(157 - 2)

## Using these estimates, run a parametric bootstrap to generate
## bootstrap replications of beta0.hat and beta1.hat
R <- 500
beta0.boot <- numeric(R)
beta1.boot <- numeric(R)
se.beta0.boot <- numeric(R)
se.beta1.boot <- numeric(R)
for(r in 1:R) {
  ysim <- beta0.hat + beta1.hat*kidney$age + sqrt(sigsq.hat/3)*rt(157, df=3)
  lm.boot <- lm(ysim ~ kidney$age)

  beta0.boot[r] <- lm.boot$coef[1]
  beta1.boot[r] <- lm.boot$coef[2]

  ## This code can be used to find the standard errors from this bootstrap sample
  sig.hatr <- summary(lm.boot)$sigma
  se.beta0.boot[r] <- sig.hatr*sqrt(summary(lm.boot)$cov.unscaled[1,1])
  se.beta1.boot[r] <- sig.hatr*sqrt(summary(lm.boot)$cov.unscaled[2,2])
}
```

- Because we have the formulas for the standard errors of β_0 and β_1, we can use studentized bootstrap confidence intervals without using the double bootstrap approach.

- Estimates of the standard error for the r^{th} bootstrap replication are

$$\begin{aligned} \hat{se}_r(\beta_0) &= \hat{\sigma}_r \sqrt{\frac{\frac{1}{n}\sum_{i=1}^n x_i^2}{S_{xx}}} \\ \hat{se}_r(\beta_1) &= \hat{\sigma}_r / \sqrt{S_{xx}} \end{aligned} \tag{10.2}$$

- These standard error estimates come from applying the formulas in (10.1) to the r^{th} bootstrap sample.

- Recall from Chapter 9 that the studentized confidence intervals are found by using the following formula.

$$\Big[T_n - se_{boot} \times \hat{K}_R^{-1}(1-\alpha/2), T_n - se_{boot} \times \hat{K}_R^{-1}(\alpha/2)\Big]$$

- `R` code to compute the studentized confidence intervals is given below:

```
## First get estimates of the standard error of our estimates
## I use the formulas for the regression standard errors, but
## we could have used a bootstrap estimate.
se.est0 <- summary(lm.kidney)$sigma*sqrt(summary(lm.boot)$cov.unscaled[1,1])
se.est1 <- summary(lm.kidney)$sigma*sqrt(summary(lm.boot)$cov.unscaled[2,2])

stu.quants0 <- quantile( (beta0.boot - beta0.hat)/se.beta0.boot, probs=c(0.025, 0.975))
stu.quants1 <- quantile( (beta1.boot - beta1.hat)/se.beta1.boot, probs=c(0.025, 0.975))
```

- The studentized bootstrap confidence intervals are then

```
## Confidence interval for beta0
c(beta0.hat - stu.quants0[2]*se.est0, beta0.hat - stu.quants0[1]*se.est0)
```

```
## (Intercept) (Intercept)
##        2.13        3.55
```

```
## Confidence interval for beta1
c(beta1.hat - stu.quants1[2]*se.est1, beta1.hat - stu.quants1[1]*se.est1)
```

```
##     age     age
## -0.0954 -0.0603
```

- Compare these studentized bootstrap confidence intervals with the confidence intervals computed under the normality assumption for the residuals:

```
confint(lm.kidney)
```

```
##               2.5 %  97.5 %
## (Intercept)  2.1497  3.5703
## age         -0.0965 -0.0607
```

- **Exercise 10.1** Using the parametric bootstrap, compute studentized bootstrap confidence for β_0 and β_1 in the kidney data example. This time, assume that $\varepsilon_i \sim \text{Normal}(0, \hat{\sigma}^2)$.

10.2.2 Nonparametric Bootstrap for Regression

- If we think of the x_i as fixed values, the Y_i in a linear regression are not i.i.d. because the means are not the same.
- This suggests that we cannot use the usual nonparametric bootstrap to construct confidence intervals for β_0 and β_1.
- However, if we also view the x_i as random, we can view the **pairs** of observations $(Y_1, x_1), \ldots, (Y_n, x_n)$ as i.i.d. observations from a bivariate distribution.
- In this case, you can also think of $\hat{\beta}_1$ as an estimate of the following quantity
$$\rho_{YX}\frac{\sigma_y}{\sigma_x}$$
where $\rho_{YX} = \text{Corr}(Y_i, x_i)$.
- In the case when (Y_i, x_i) are bivariate normal, the conditional expectation of Y_i given x_i has the linear regression structure:
$$E(Y_i|x_i) = \beta_0 + \beta_1 x_i,$$
where $\beta_0 = \mu_y - \rho_{YX}\frac{\sigma_y \mu_x}{\sigma_x}$ and $\beta_1 = \rho_{YX}\frac{\sigma_y}{\sigma_x}$.
- So, even if the linear model is not exactly true, our estimate and confidence interval still has a clear interpretation.
- If the linear model assumption is true, the true standard error of $\hat{\beta}_0$ and $\hat{\beta}_1$ will be slightly different than the formulas shown in (10.1). Nevertheless, (10.1) can be thought of as consistent estimates of the true standard error.

- If we are thinking of the observations $(Y_1, x_1), \ldots, (Y_n, x_n)$ as i.i.d. pairs, we can use the nonparametric bootstrap by just subsampling pairs of observations.

- So, to generate bootstrap replications $\hat{\beta}^*_{0,r}$, $\hat{\beta}^*_{1,r}$ for $\hat{\beta}_0$ and $\hat{\beta}_1$, we just use the following procedure

- For $r = 1, \ldots, R$:

  - Draw a sample of size n: $((Y_1^*, x_1^*), \ldots, (Y_n^*, x_n^*))$ by sampling with replacement from the original data.
  - Compute $\hat{\beta}^*_{0,r}$ and $\hat{\beta}^*_{1,r}$ from this bootstrap sample.

- `R` code for generating these bootstrap replications for the `kidney` data is below:

```
R <- 500
beta0.boot.np <- numeric(R)
beta1.boot.np <- numeric(R)
se.beta0.boot.np <- numeric(R)
se.beta1.boot.np <- numeric(R)
for(r in 1:R) {
  subsamp.ind <- sample(1:157, size=157, replace=TRUE)
  kidney.tmp <- kidney[subsamp.ind,]
  lm.boot <- lm(tot ~ age, data=kidney.tmp)

  beta0.boot.np[r] <- lm.boot$coef[1]
  beta1.boot.np[r] <- lm.boot$coef[2]

  ## This code can be used to find the standard errors from this bootstrap sample
  sig.hatr <- summary(lm.boot)$sigma
  se.beta0.boot.np[r] <- sig.hatr*sqrt(summary(lm.boot)$cov.unscaled[1,1])
  se.beta1.boot.np[r] <- sig.hatr*sqrt(summary(lm.boot)$cov.unscaled[2,2])
}
```

- To find the studentized confidence intervals for this nonparametric bootstrap, we can use the following code:

```
se.est0 <- summary(lm.kidney)$sigma*sqrt(summary(lm.boot)$cov.unscaled[1,1])
se.est1 <- summary(lm.kidney)$sigma*sqrt(summary(lm.boot)$cov.unscaled[2,2])

stu.quants0.np <- quantile( (beta0.boot.np - beta0.hat)/se.beta0.boot.np,
                           probs=c(0.025, 0.975))
stu.quants1.np <- quantile( (beta1.boot.np - beta1.hat)/se.beta1.boot.np,
                           probs=c(0.025, 0.975))
```

- The studentized bootstrap confidence intervals for β_0 and β_1 are then

```
## Confidence interval for beta0
c(beta0.hat - stu.quants0.np[2]*se.est0, beta0.hat - stu.quants0.np[1]*se.est0)
```

```
## (Intercept) (Intercept)
##        2.10        3.62
```

```
## Confidence interval for beta1
c(beta1.hat - stu.quants1.np[2]*se.est1, beta1.hat - stu.quants1.np[1]*se.est1)
```

```
##     age     age
## -0.0974 -0.0587
```

- **Exercise 10.2** Another way of using the bootstrap in a regression context is to resample the residuals from the fitted regression model. Speficially, we first fit the linear regression model and compute residuals $\hat{e}_1, \dots, \hat{e}_n$ via

$$\hat{e}_i = Y_i - \hat{\beta}_0 - \hat{\beta}_1 x_i$$

One then generates a bootstrap sample by first subsampling $(\hat{e}_1^*, \dots, \hat{e}_n^*)$ from the vector of "original" residuals $(\hat{e}_1, \dots, \hat{e}_n)$ and then setting $Y_i^* = \hat{\beta}_0 + \hat{\beta}_1 x_i + \hat{e}_i^*$. You then compute the bootstrap replications $\hat{\beta}_{0,r}^*$ and $\hat{\beta}_{1,r}^*$ by fitting a linear regression with data: $(Y_1^*, x_1), \dots, (Y_n^*, x_n)$.

Using the kidney data, try using this procedure to construct 95% bootstrap confidence intervals for β_0 and β_1.

Regression with more than 1 covariate

- If we have more than one covariate in our model, for example,

$$Y_i = \beta_0 + \beta_1 x_{i1} + \dots + \beta_p x_{ip} + \varepsilon_i,$$

the bootstrap works essentially the same as for the case with a single covariate.

- For the parametric bootstrap with a Normal residual distribution, you would just simulate $Y_i^* \sim \text{Normal}(\hat{\beta}_0 + \hat{\beta}_1 x_{i1} + \dots + \hat{\beta}_p x_{ip}, \hat{\sigma}^2)$.
- For the nonparametric bootstrap, you would subsample pairs $(Y_1^*, x_1^*), \dots, (Y_n^*, x_n^*)$ as described before, and compute your regression coefficients $\hat{\beta}_{0,r}^*, \hat{\beta}_{1,r}^*, \dots, \hat{\beta}_{p,r}^*$ from this bootstrap sample.

10.3 Pointwise Confidence Intervals for a Density Function

- Recall that a kernel density estimate of an unknown probability density $f(x)$ has the form

$$\hat{f}_{h_n}(x) = \frac{1}{nh_n}\sum_{i=1}^{n} K\Big(\frac{x - X_i}{h_n}\Big)$$

- We cannot naively apply the Central Limit Theorem, because h_n is changing as $n \longrightarrow \infty$.
- Nevertheless, you can show (see, e.g., Tsybakov (2008)) that

$$\sqrt{nh_n}\Big(\hat{f}_{h_n}(x) - E\{\hat{f}_{h_n}(x)\}\Big) \longrightarrow \text{Normal}(0, \kappa_2(K)f(x))$$

provided that $h_n \longrightarrow 0$ and $nh_n \longrightarrow \infty$. Here, $\kappa_2(K) = \int_{-\infty}^{\infty} K^2(u)du$.

- This suggests that a standard error estimate for $\hat{f}_{h_n}(x)$ is $\sqrt{\kappa_2(K)\hat{f}_{h_n}(x)/nh_n}$ and a 95% confidence interval for $E\{\hat{f}_{h_n}(x)\}$ is

$$\left[\hat{f}_{h_n}(x) - 1.96 \times \sqrt{\frac{\kappa_2(K)\hat{f}_{h_n}(x)}{nh_n}}, \hat{f}_{h_n}(x) + 1.96 \times \sqrt{\frac{\kappa_2(K)\hat{f}_{h_n}(x)}{nh_n}}\right]$$

- Notice that this is a confidence interval for $E\{\hat{f}_{h_n}(x)\}$ rather than $f(x)$.
- So, you can roughly think of this as a confidence interval for a smoothed version of $f(x)$ at x:

$$E\{\hat{f}_{h_n}(x)\} = \frac{1}{h_n}\int_{-\infty}^{\infty} K\Big(\frac{x - t}{h_n}\Big)f(t)dt$$

- Notice also that this is a pointwise confidence interval. It is not a confidence band.
- Methods for computing "bias-corrected" confidence intervals for $f(x)$ are discussed, for example, in Chen (2017).

- To get a bootstrap estimate of the standard deviation of $\hat{f}_{h_n}(x)$, we can use the usual steps.

- For $r = 1, \dots, R$:
  - Draw a sample of size n: $(X_1^*, \dots, X_n^*)$ by sampling with replacement from $\mathbf{X}$.
  - Compute $T_{n,r}^* = \frac{1}{nh_n}\sum_{i=1}^n K(\frac{x - X_i^*}{h_n})$.

Then, compute the estimated standard error:

$$\hat{se}_{boot} = \Big[\frac{1}{R-1}\sum_{r=1}^{R}\Big(T_{n,r}^* - \frac{1}{R}\sum_{r=1}^{R} T_{n,r}^*)^2\Big]^{1/2} \tag{10.3}$$

- `R` code to compute these standard error estimates for the `sysBP` variable from the `framingham` dataset is given below

```
framingham <- read.csv("~/Documents/STAT685Notes/Data/framingham.csv")
R <- 500
BootMat <- matrix(0, nrow=R, ncol=4)
for(r in 1:R) {
    xx.boot <- sample(framingham$sysBP, size=length(framingham$sysBP), replace=TRUE)
    kk.boot <- density(xx.boot)
    tmp <- approxfun(kk.boot$x, kk.boot$y)
    BootMat[r,] <- c(tmp(100), tmp(125), tmp(150), tmp(175))
}
bb <- apply(BootMat, 2, sd)
```

10.4 When can the Bootstrap Fail?

- While the bootstrap is very automatic and could be used to construct confidence intervals in nearly any situation, these bootstrap confidence intervals may fail to give the correct coverage in some situations.
- A few situations in which the bootstrap can fail include:
  - If we are interested in estimating a parameter θ and the support $\{x : f_\theta(x) > 0\}$ of the density function depends on θ.
  - If there are parameter constraints and the true value of the parameter lies on the boundary of the parameter space. For example, we estimate θ subject to the constraint that $\theta \geq 0$, and the true value of θ is zero.
  - If $T_n = g(\bar{X})$ and $g'(\mu) = 0$ where $\mu = E(X_1)$.
  - No finite mean. If $E(|X_1|)$ is not finite, then the bootstrap may not work well.

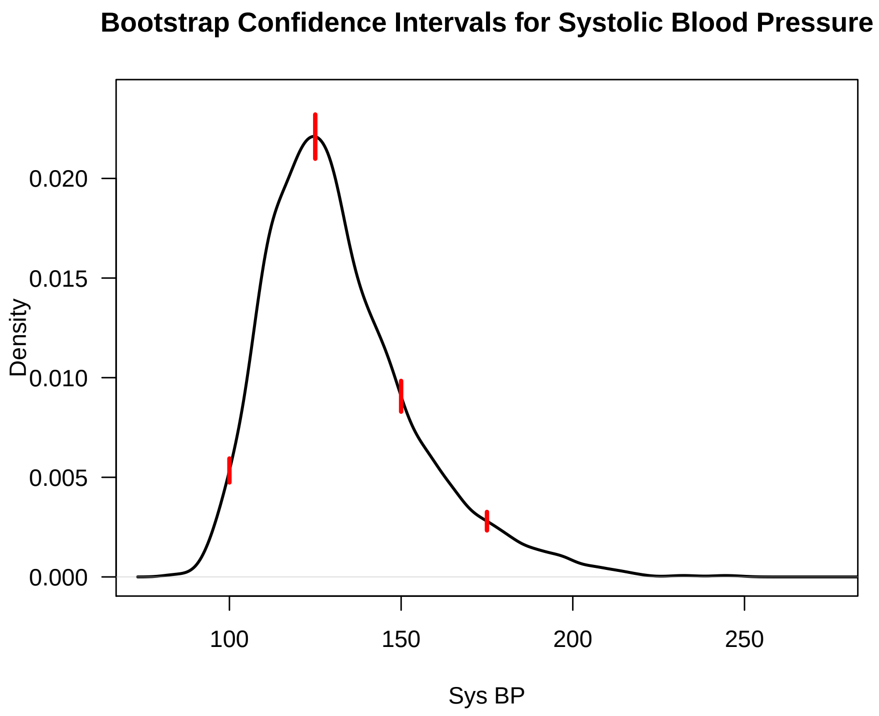


Figure 10.1: Bootstrap confidence intervals for the density function at the points x=100, 125, 150, 175

10.4.1 Example: The Shifted Exponential Distribution

- Let us consider observations $X_1, \dots, X_n$ that follow the shifted exponential distribution whose density function is

$$f(x) = \begin{cases} \lambda e^{-\lambda(x-\eta)} & \text{if } x \geq \eta \\ 0 & \text{otherwise} \end{cases}$$

  where $\lambda > 0$ and $\eta > 0$.

- The maximum likelihood estimates of λ and η are

$$\hat{\lambda} = \frac{1}{\bar{X}} - X_{(1)} \qquad \hat{\eta} = X_{(1)}$$

  where $X_{(1)} = \min\{X_1, \dots, X_n\}$ is the smallest observation.

- Notice that this is an example where the support of the density function depends on the parameter η.

- Suppose we use the bootstrap to construct confidence intervals for λ and η. What will happen?

- Let us consider an example where we have i.i.d. data $X_1, \dots, X_n$ that follow a shifted Exponential distribution with $\lambda = 1/3$ and $\eta = 2$.

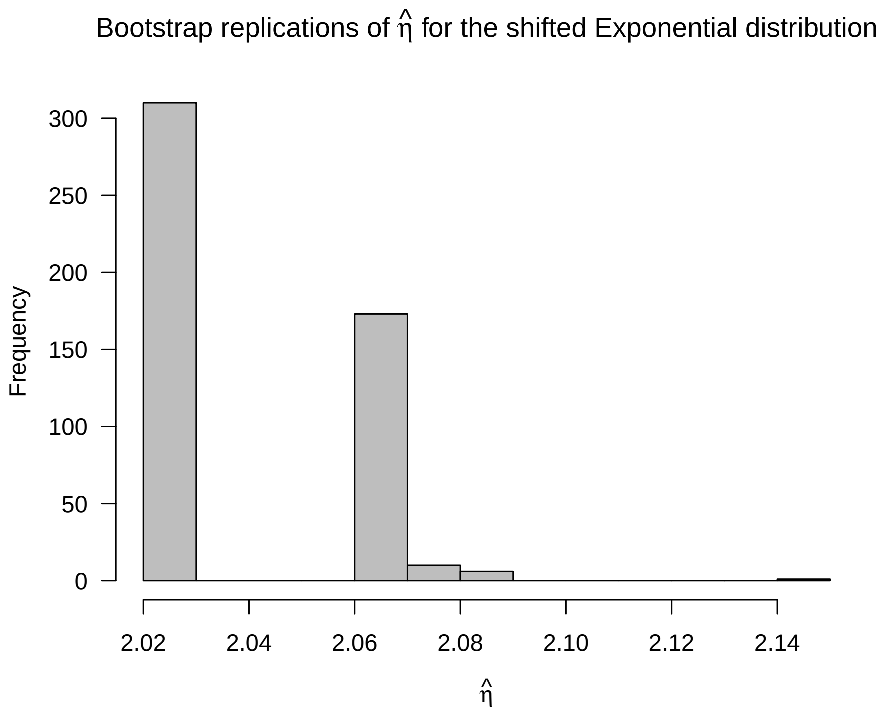
- The following code can estimate the coverage proportion of a bootstrap confidence interval for η:

```
n <- 200
R <- 500
eta.true <- 2

nreps <- 500
Cover.bootsd.ci <- numeric(nreps)
for(k in 1:nreps)  {
  ## Step 1: Generate the Data and compute the estimate of eta
  xx <- 2 + rexp(n, rate=1/3)
  eta.hat <- min(xx)

  ## Step 2: Find bootstrap confidence intervals using R bootstrap replications
  eta.boot <- numeric(R)
  for(r in 1:R)   {
    boot.xx <- sample(xx, size=n, replace = TRUE)
    eta.boot[r] <- min(boot.xx)
  }
  boot.ci.sd <- c(eta.hat - 1.96*sd(eta.boot), eta.hat + 1.96*sd(eta.boot))

  ## Step 3: Record if the true parameter is covered or not:
  Cover.bootsd.ci[k] <- ifelse(boot.ci.sd[1] < eta.true & boot.ci.sd[2] >= eta.true,
                               1, 0)
}
```

- The estimated coverage for this bootstrap confidence interval is

```
mean(Cover.bootsd.ci)
```

```
## [1] 0.816
```



10.5 The Jackknife

- The jackknife is also a nonparametric method for estimating standard errors.

- Like the bootstrap, the jackknife also uses the idea of looking at multiple subsets of the data.
- Also like the bootstrap, the jackknife is completely automatic in the sense that we only need to be able to compute our statistic of interest, and we do not need to do any formal calculations to find the standard error.
- While the jackknife was actually developed before the bootstrap, it is used much less than the bootstrap is in applications - at least in the context of finding confidence intervals.

- We will define $\mathbf{X}_{-i}$ to be the vector of observations that has the i^{th} observation deleted:

$$\mathbf{X}_{(-i)} = (X_1, \ldots, X_{i-1}, X_{i+1}, \ldots, X_n)$$

- Define $T_{n,(-i)}$ to be the value of the statistic T_n when using data which has the i^{th} observation removed

$$T_{n,(-i)} = h(X_1, \ldots, X_{i-1}, X_{i+1}, \ldots, X_n)$$

- The jackknife estimate of the standard error of T_n is

$$\hat{se}_{jack} = \Big[\frac{n-1}{n}\sum_{i=1}^{n}(T_{n,(-i)} - \bar{T}_{n,jack})^2\Big]^{1/2},$$

where $\bar{T}_{n,jack} = \frac{1}{n}\sum_{i=1}^{n} T_{n,(-i)}$.

- An advantage of the jackknife is that, like the bootstrap, it does not make any particular parametric assumptions about the distribution of the data.
- However, the jackknife is more dependent on a smoothness assumption (that is smoothness across slightly perturbed datasets) than the bootstrap. An example of this is the sample median where, if we delete one observation, the sample median has a different definition due to the sample size being even vs. odd.

Part IV

Nonparametric Regression: Part I

Chapter 11

Kernel Regression and Local Regression

11.1 Introduction

- In regression we are interested in characterizing, in some way, the relationship between a collection of responses $Y_1, \ldots, Y_n$ and covariate vectors $(\mathbf{x}_1, \ldots, \mathbf{x}_n)$.

- Linear regression is one way of approaching this problem. This assumes the expectation of Y_i can be expressed as a linear combination of the covariates:
$$E(Y_i|\mathbf{x}_i) = \beta_0 + \sum_{j=1}^{p} x_{ij}\beta_j$$

- More generally, we can consider the following model
$$Y_i = m(\mathbf{x}_i) + \varepsilon_i$$
  - $m(\mathbf{x}_i)$ - the "mean function" or "regression function"
  - $\mathbf{x}_i = (x_{i1}, \ldots, x_{ip})$ - the i^{th} covariate vector

- The residuals $\varepsilon_1, \ldots, \varepsilon_n$ are assumed to be i.i.d. and have mean zero.

- In a nonparametric approach, we will try to estimate $m(\mathbf{x})$ without making any strong assumptions about the form of $m(\mathbf{x})$.

- The regression function $m(\mathbf{x})$ can be thought of as the function which returns the expectation of Y_i given that $\mathbf{x}_i = \mathbf{x}$

$$m(\mathbf{x}) = E(Y_i|\mathbf{x}_i = \mathbf{x}) \tag{11.1}$$

- Let
  - $f_{Y|X}(y|\mathbf{x})$ denote the conditional density of Y_i given $\mathbf{x}_i$.
  - $f_{Y,X}(y, \mathbf{x})$ denote the joint density of $(Y_i, \mathbf{x}_i)$
  - $f_X(\mathbf{x})$ denote the density of $\mathbf{x}_i$

- We can express the regression function as

$$m(\mathbf{x}) = \int_{-\infty}^{\infty} y f_{Y|X}(y|\mathbf{x}) dy = \frac{\int y f_{Y,X}(y, \mathbf{x}) dy}{f_X(\mathbf{x})}$$

11.2 Kernel Regression

- In this section, we will assume that the covariates are univariate. That is, $p = 1$ and $\mathbf{x}_i = x_i$ where x_i is a real number.

11.2.1 The Regressogram

- The regressogram is an estimate of the mean function $m(x)$ which is has many similarities in its construction to the histogram.

- Similar to how we constructed the histogram, let us think about an estimate $m(x)$ that will be constant within each of a series of bins $B_1, \dots, B_{D_n}$

$$\begin{aligned} B_1 &= [x_0, x_0 + h_n) \\ B_2 &= [x_0 + h_n, x_0 + 2h_n) \\ &\vdots \\ B_{D_n} &= [x_0 + (D_n - 1)h_n, x_0 + D_n h_n) \end{aligned}$$

- Suppose we want to estimate $m(x)$, where x belongs to the k^{th} bin. A direct estimate of this is the average of the $Y_i's$ among those $x_i's$ which fall into the k^{th} bin.

- Specifically, if $x \in B_k$, then we estimate $m(x)$ with

$$\hat{m}_{h_n}^R(x) = \frac{\sum_{i=1}^n Y_i I(x_i \in B_k)}{\sum_{i=1}^n I(x_i \in B_k)} = \frac{1}{n_{k,h_n}} \sum_{i=1}^n Y_i I(x_i \in B_k),$$

  where n_{k,h_n} is the number of x_i that fall into the k^{th} bin when using bin width h_n.

- The estimate $\hat{m}^R_{h_n}(x)$ of the regression function is called the **regressogram**.

- The intuition for this estimate is: if $x \in B_k$, then taking an average of the reponses for x_i in a small bin containing x should give us a reasonable approximation for the expectation of Y_i given that $x_i = x$.

- Another way of looking at the regressogram is to note that if we think of the x_i as random variables, then for $x \in B_k$

$$\begin{aligned} E\Big\{\frac{1}{n}\sum_{i=1}^n Y_i I(x_i \in B_k)\Big\} &= E\Big\{Y_1 I(x_1 \in B_k)\Big\} \\ &= \int_{-\infty}^{\infty}\int_{x_0+(k-1)h_n}^{x_0+kh_n} y f_{Y,X}(y,t)dtdy \\ &\approx h_n \int_{-\infty}^{\infty} y f_{Y,X}(y,x)dy \qquad (11.2) \end{aligned}$$

  and, similarly,

$$\begin{aligned} E\Big\{\frac{1}{n}\sum_{i=1}^n I(x_i \in B_k)\Big\} &= E\Big\{I(x_1 \in B_k)\Big\} \\ &= \int_{x_0+(k-1)h_n}^{x_0+kh_n} f_X(t)dt \\ &\approx h_n f_X(x) \qquad (11.3) \end{aligned}$$

- Equations (11.2) and (11.3) suggest that $\hat{m}^R_{h_n}(x)$ should be a reasonable estimate of the ratio

$$\int_{-\infty}^{\infty} y f_{Y,X}(y,x)dy \Big/ f_X(x)$$

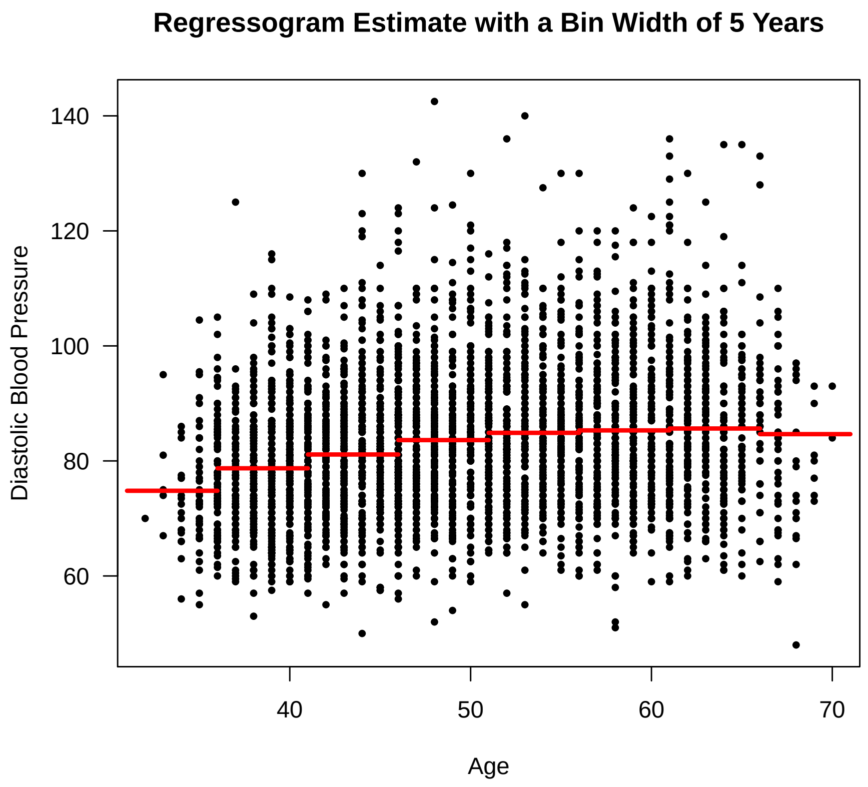
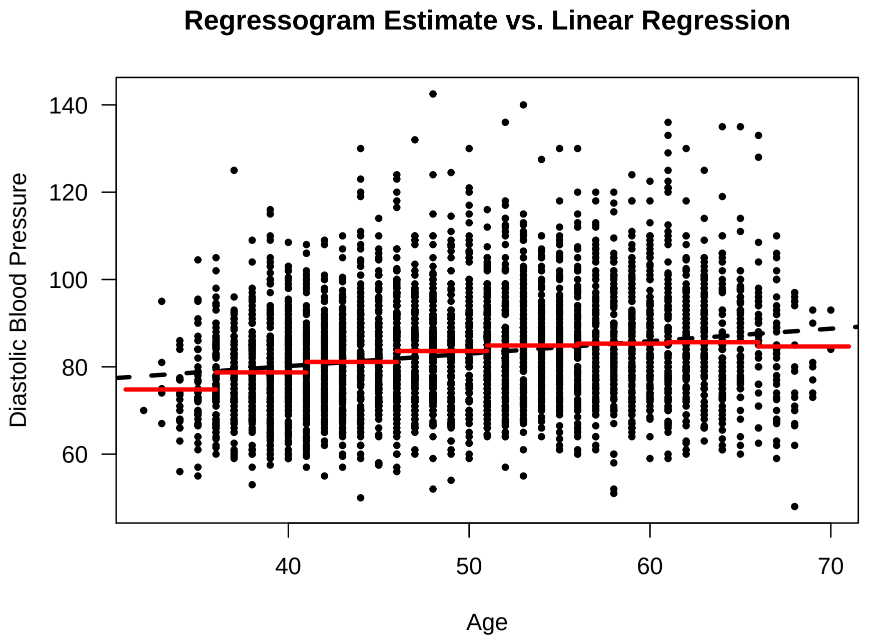


Figure 11.1: Framingham Data. Regressogram estimate for a regression model with diastolic blood pressure as the response and age as the covariate. Ages from 31-71 were separated into bins of width 5 years.



- **Exercise 11.1** Let

  $$\hat{\mathbf{m}} = (\hat{m}_{h_n}^R(x_1), \ldots, \hat{m}_{h_n}^R(x_n))$$

  denote the vector of "fitted values" from a regressogram estimate that has D_n bins. If $\mathbf{Y} = (Y_1, \ldots, Y_n)$, show that you can express $\hat{\mathbf{m}}$ as

  $$\hat{\mathbf{m}} = \mathbf{A}\mathbf{Y},$$

  for an appropriately chosen $n \times n$ matrix $\mathbf{A}$. What is the value of $\text{tr}(\mathbf{A})$?

11.2.2 The Local Average Estimator

- The regressogram can be thought of as a regression analogue of the histogram.
- The local average estimator can be thought of as a regression analogue of the "box-type" density estimator that we described in Chapter 8.

- For each point x, we are going to use a regression function estimate which has a bin "centered" at x.
- Specifically, for each x, we will form a bin of width $2h_n$ around x and compute the mean of the Y_i among those observations where the x_i fall into this bin.
- In other words, we are computing an average of the Y_i in a small region around x.
- The local average estimator $\hat{m}_{h_n}^{loc}(x)$ at x is defined as:

$$\begin{aligned}\hat{m}_{h_n}^{loc}(x) &= \frac{\sum_{i=1}^{n} Y_i I(x - h_n < x_i < x + h_n)}{\sum_{i=1}^{n} I(x - h_n < x_i < x + h_n)} \\ &= \frac{1}{n_{h_n}(x)} \sum_{i=1}^{n} Y_i I(x - h_n < x_i < x + h_n)\end{aligned}$$

  where $n_{h_n}(x) = \sum_{i=1}^{n} I(x - h_n < x_i < x + h_n)$.
- The local average estimator does not need to have a constant value within each of a few pre-specified bins.

- We can also express the local average estimator in the following way:

$$\hat{m}_{h_n}^{loc}(x) = \frac{\sum_{i=1}^{n} Y_i w\Big(\frac{x - X_i}{h_n}\Big)}{\sum_{i=1}^{n} w\Big(\frac{x - X_i}{h_n}\Big)}, \tag{11.4}$$

  where $w(t)$ is the "box" function defined as

$$w(t) = \begin{cases} \frac{1}{2} & \text{if } |t| < 1 \\ 0 & \text{otherwise} \end{cases}$$

- While a local average estimate will not be a "step function" like the regressogram, the local average estimate will typically be non-smooth and have a jagged appearance.

- Like kernel density estimation, there is a bias/variance tradeoff to the choice of h_n.

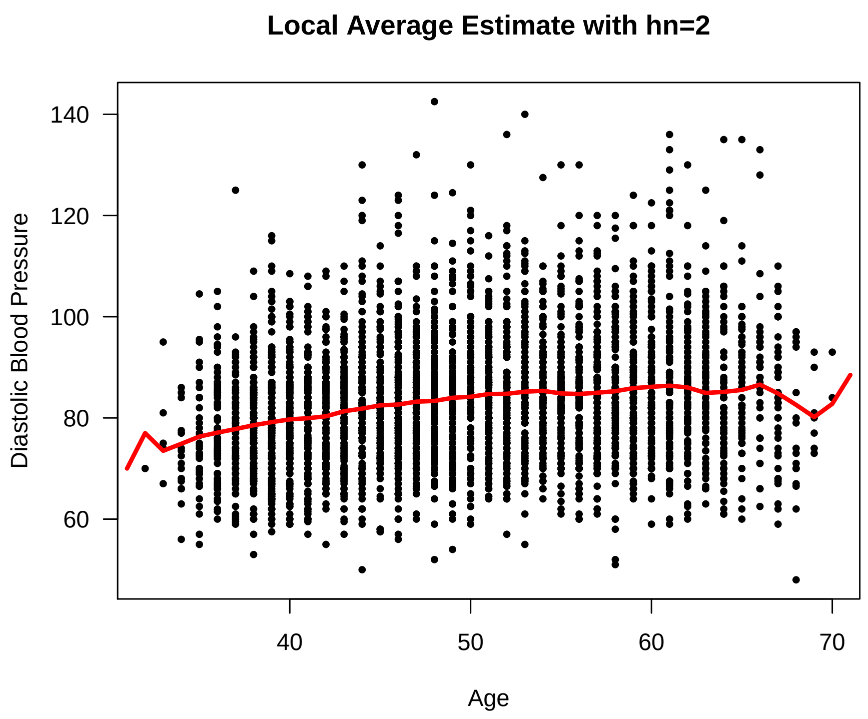
- Smaller values of h_n usually imply higher variance because you will be taking an average over a relatively small number of observations.

- Larger values of h_n usually imply higher bias because you will be esitmating $m(x)$ by averaging over a wide range of x_i values, and $m(x)$ could vary substantially over this range of x_i values.

- Our experience in Chapter 8 suggests that we can get a smoother estimate of the regression if we simply replace the "box function" $w(t)$ in (11.4) with a smoother kernel function $K(t)$.

- R code for computing a local average estimate $\hat{m}_2^{loc}(x)$ at the points $x = 31, 32, 33, ..., 71$ is given below

```r
xseq <- seq(31, 71, by=1)
hn <- 2
nx <- length(xseq)
m.hat.loc <- numeric(nx)
for(k in 1:nx) {
    in.bin <- framingham$age > xseq[k] - hn & framingham$age < xseq[k] + hn
    m.hat.loc[k] <- mean(framingham$diaBP[in.bin])
}

plot(framingham$age, framingham$diaBP, las=1, ylab="Diastolic Blood Pressure",
     xlab="Age", main="Local Average Estimate with hn=2", type="n")
points(framingham$age, framingham$diaBP, pch=16, cex=0.7)
lines(xseq, m.hat.loc, lwd=3, col="red")
```



- Let's also look at a local average estimate of the regression function for the bone mineral density dataset.

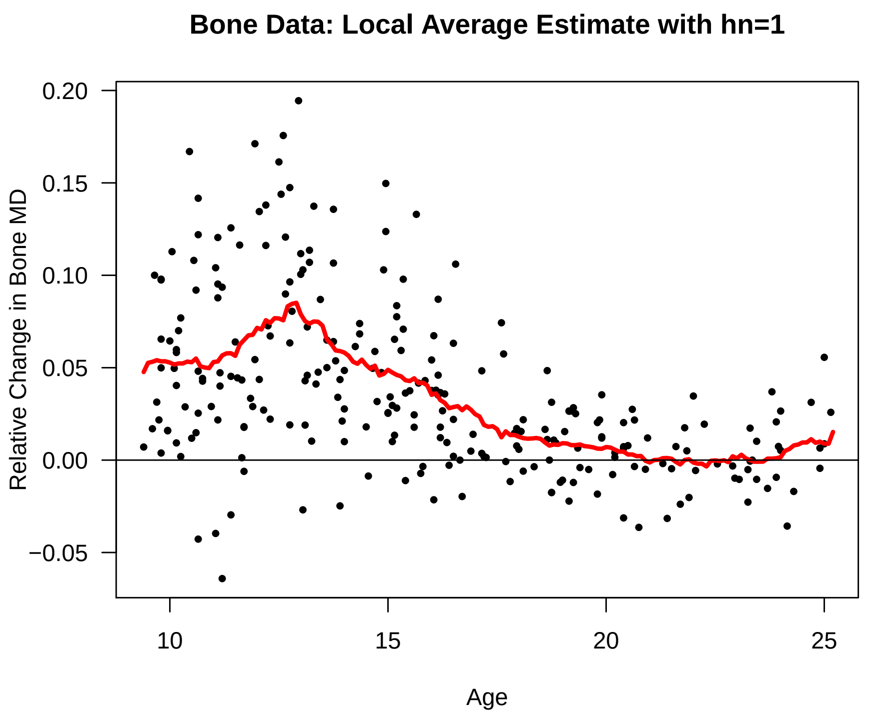
- The responses in this dataset are relative changes in the bone mineral density of adolescents.

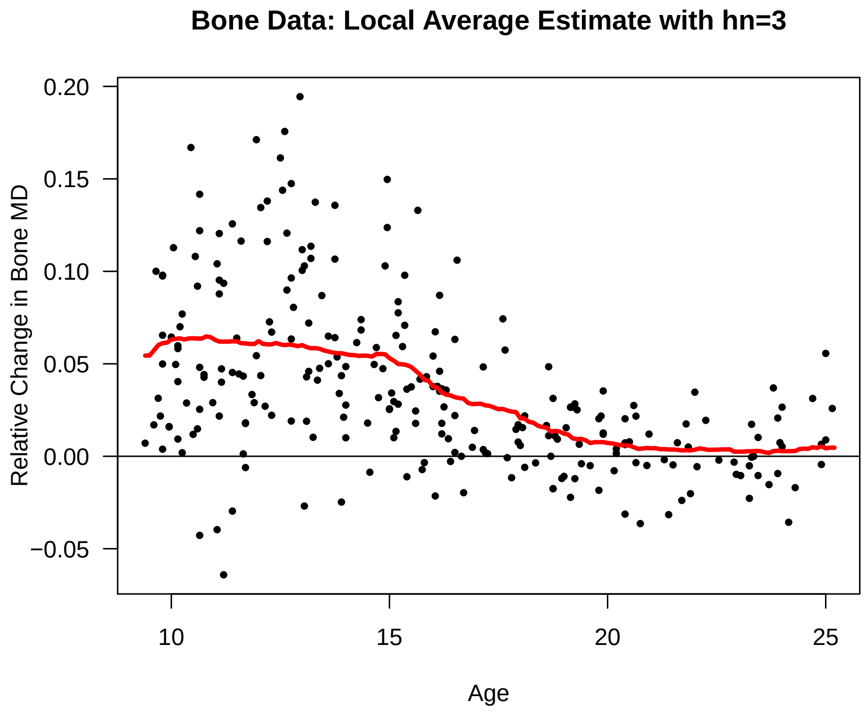
- Specifically, reponses Y_i and covariates x_i are defined as

$$\begin{aligned} Y_i &= \frac{\text{Mineral Density at Visit } 2_i - \text{Mineral Density at Visit } 1_i}{\frac{1}{2}(\text{Mineral Density at Visit } 2_i + \text{Mineral Density at Visit } 1_i)} \\ x_i &= \frac{1}{2}(\text{Age at Visit } 2_i + \text{Age at Visit } 1_i) \end{aligned}$$

```
tmp <- read.table("https://web.stanford.edu/~hastie/ElemStatLearn/datasets/bone.data",
                  header=TRUE)
bonedat <- tmp[!duplicated(tmp$idnum),]  ## only keep the first observation of a perso
```

```
xseq <- seq(9.4, 25.2, by=.1)
hn <- 1
nx <- length(xseq)
m.hat.loc <- numeric(nx)
for(k in 1:nx) {
    in.bin <- bonedat$age > xseq[k] - hn & bonedat$age < xseq[k] + hn
    m.hat.loc[k] <- mean(bonedat$spnbmd[in.bin])
}

plot(bonedat$age, bonedat$spnbmd, las=1, ylab="Relative Change in Bone MD",
     xlab="Age", main="Bone Data: Local Average Estimate with hn=1", type="n")
points(bonedat$age, bonedat$spnbmd, pch=16, cex=0.7)
lines(xseq, m.hat.loc, lwd=3, col="red")
abline(0, 0)
```





11.2.3 k-Nearest Neighbor (k-NN) Regression

- k-nearest neighbor regression is fairly similar to the local average estimator of the regression function.
- With k-NN, we still estimate the regression function at a particular point by taking a type of local average around this point.
- However, k-NN takes the average over the k "nearest observations" to x rather than taking an average over all the observations which fall into a bin centered at x.

- The k-NN estimator of the regression function $\hat{m}_k^{kNN}(x)$ is defined as

$$\hat{m}_k^{kNN}(x) = \frac{1}{k} \sum_{i=1}^{n} y_i I(x_i \in N_k(x))$$

- Here, $N_k(x)$ is defined as the set of the k $x_i's$ which are closest to x. That is, $N_k(x)$ is the set of the k "nearest neighbors" to x.
- Mathematically, if we define

$$d_i(x) = |x_i - x|$$

  and order them so that $d_{(1)}(x) \leq d_{(2)}(x) \leq ... \leq d_{(n)}(x)$. Then, the k nearest neighbors of x would be those observations which correspond to the $d_{(1)}(x)$ through $d_{(k)}(x)$.

- Like the local average estimator, increasing the value of k will increase the bias of the k-NN regression function estimate while decreasing the value of k will increase the variance of the k-NN regression function estimate.

- **Exercise 11.2** Suppose $n = 6$ and that we have the following covariate values and responses

$$\begin{aligned}(x_1, x_2, x_3, x_4, x_5, x_6) &= (1/7, 2/7, 3/7, 4/7, 5/7, 6/7) \\ (Y_1, Y_2, Y_3, Y_4, Y_5, Y_6) &= (1.4, 0.7, 1.1, 1.3, 0.9, 1.7)\end{aligned}$$

  - Compute the local average estimate of the regression function at $x = 0.25$ and $x = 0.75$ assuming that $h_n = 1/2$.
  - Compute the k nearest neighbors estimate of the regression function at $x = 0.25$ and $x = 0.75$ assuming that $k = 2$.

11.2.4 The Nadaraya-Watson Estimator

- The Nadaraya-Watson estimator $\hat{m}_{h_n}^{NW}$ of the regression function with bandwidth h_n is defined as

$$\hat{m}_{h_n}^{NW}(x) = \frac{\sum_{i=1}^{n} Y_i K\left(\frac{x-x_i}{h_n}\right)}{\sum_{i=1}^{n} K\left(\frac{x-x_i}{h_n}\right)}$$

- The Nadaraya-Watson estimator has the same basic form as the local average estimator. We have just replaced the "box" function $w(t)$ with the kernel function $K(t)$.

- You can think of $\hat{m}_{h_n}^{NW}(x)$ as a weighted average of the Y_i. That is,

$$\hat{m}_{h_n}^{NW}(x) = \sum_{i=1}^{n} a_i(x) Y_i$$

- The bandwidth h_n can also be referred to as the "smoothing parameter" since its value affects how smooth the fitted regression curve appears.

- The weights $a_1(x), \dots, a_n(x)$, in this case, are defined as

$$a_i(x) = \frac{K(\frac{x - x_i}{h_n})}{\sum_{i=1}^{n} K(\frac{x - x_i}{h_n})}$$

So, we are using weights which are larger the closer you are to x.

- The Nadaraya-Watson estimator suffers from two main drawbacks. These are **design bias** and **boundary bias**.

- Design bias refers to the effect of the spacing of the x_i on the performance of the Nadaraya-Watson estimator.

- Boundary bias refers to the performance of the Nadaraya-Watson estimator near the smallest and largest x_i.

- If we assume that the x_i are random and have probability density $f_X(x)$, then it can be shown that the mean-squared error of the Nadaraya-Watson estimator at a particular point x has the following approximation

$$\begin{aligned} \text{MSE}(x) &= E\Big[\{m(x) - \hat{m}_{h_n}^{NW}(x)\}^2\Big] \\ &\approx \frac{h_n^4 \mu_2^2(K)}{4}\left\{m''(x) + \frac{2m'(x)f_X'(x)}{f_X(x)}\right\}^2 + \frac{\sigma^2}{nh_n f_X(x)}, \end{aligned}$$

where $\mu_2(K) = \int_{-\infty}^{\infty} u^2 K(u)du$ and $\kappa_2(K) = \int_{-\infty}^{\infty} K^2(u)du$.

- The term $2m'(x)f_X'(x)/f_X(x)$ is referred to as the design bias. Notice that this should be zero if the x_i are drawn from a Uniform distribution. In other words, if the x_i are roughly equally spaced, then the design bias should be small.

The Nadaraya-Watson estimator in R

- The Nadaraya-Watson estimator can be computed in `R` with the `ksmooth` function.

```
ksmooth(x, y, kernel, bandwidth, x.points, ...)
```

-
    - `x` - vector of covariate values
    - `y` - vector of responses
    - `kernel` - choice of kernel function; default is `box`; use `normal` if you want a Gaussian kernel
    - `bandwidth` - value of the bandwidth; default is 0.5
    - `x.points` - points at which to estimate the regression function; default is to use n equally spaced points.

- The `x` vector from the fitted `ksmooth` object will be the vector of points at which the regression function is estimated. The `y` vector from the fitted `ksmooth` object will be a vector containing the estimated values of the regression function.

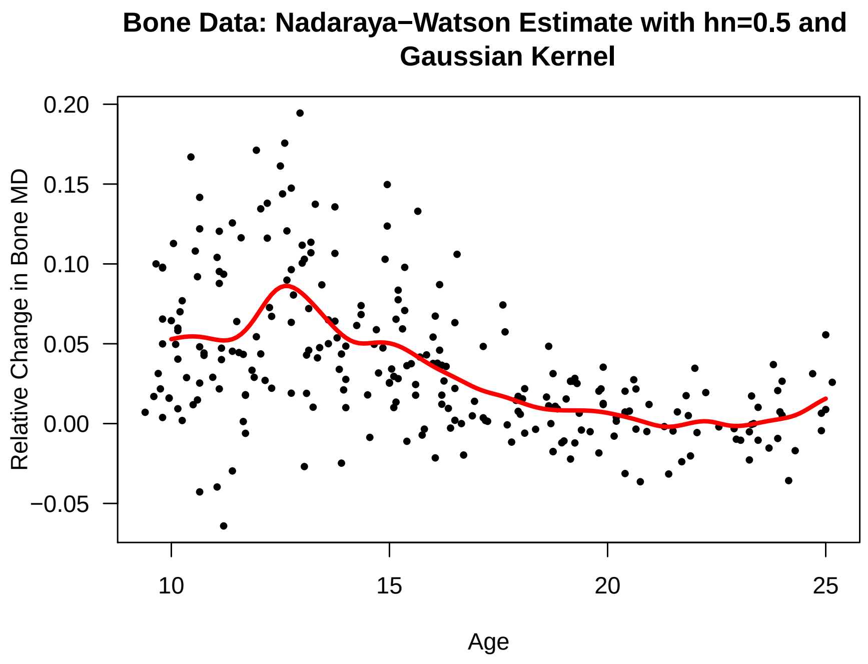
- Note that the bandwidth used by this function for the Gaussian kernel is approximately 2.7 times smaller than the bandwith in our definition of the Gaussian kernel.

- If you wanted to write your own function that computed the Nadaraya-Watson estimate at a vector of desired points $x.points = (t_1, \ldots, t_q)$, you could use something like

```
MyNWEst <- function(x, y, bandwidth, x.points) {
    q <- length(x.points)
    nw.est <- numeric(q)
    for(k in 1:q) {
        ww <- dnorm(x.points[k], mean=x, sd=bandwidth)
        nw.est[k] <- sum(ww*y)/sum(ww)
    }
    return(nw.est)
}
```

- To compute the Nadraya-Watson estimate at a set of equally spaced of points from 10 to 25 using bandwidth 0.5 and plot the result, you could use the following code:

```
xseq <- seq(10, 25, by=.1)
bone.nwest <- ksmooth(x=bonedat$age, y=bonedat$spnbmd, kernel="normal",
                      bandwidth=2.7*0.5, x.points=xseq)

plot(bonedat$age, bonedat$spnbmd, las=1, ylab="Relative Change in Bone MD",
     xlab="Age", main="Bone Data: Nadaraya-Watson Estimate with hn=0.5 and
     Gaussian Kernel", type="n")
points(bonedat$age, bonedat$spnbmd, pch=16, cex=0.7)
lines(bone.nwest$x, bone.nwest$y, lwd=3, col="red")
```



```
## Note that bone.nwest$x should equal xseq
```

11.3 Local Linear Regression

11.3.1 Definition

- You can think of both the regressogram and the local average as methods which fit local intercept models.
- For the regressogram, we fit an intercept model (that is a flat line curve) within a small bin that contains x.

- For the local average estimator, we fit an intercept model for a small bin around x.
- The Nadaraya-Watson estimator can be thought of as just smoothing out the local intercept approach of the local average estimator.

- Instead of fitting local intercept models, we could fit local linear models that have an intercept and a slope term.
- To be specific, suppose we estimated the regression function at x by fitting a linear model using only data from the x_i that fell into the bin $(x-h_n, x+h_n)$.
- In this case, we would first fit the linear model $\hat{s}_x(x_i) = \hat{\beta}_{0x} + \hat{\beta}_{1x}(x_i - x)$ where $\hat{\beta}_{0x}$, $\hat{\beta}_{1x}$ solved the following local least-squares problem

$$\hat{\beta}_{0x}, \hat{\beta}_{1x} \text{ minimize:} \qquad \sum_{i=1}^{n}\{Y_i - \beta_{0x} - \beta_{1x}(x_i - x)\}^2 I(x - h_n < x_i < x + h_n) \tag{11.5}$$

- Then, we would estimate $m(x)$ by using the value of $\hat{s}_x(\cdot)$ at x. That is, $\hat{s}_x(x) = \hat{\beta}_{0x}$.

- **Local Linear Regression** uses the same idea as (11.5), but replaces the indicator function $I(x - h_n < x_i < x + h_n)$ with a smooth kernel function.
- So, the local linear regression estimate of the regression function at x is

$$\begin{aligned}\hat{m}_{h_n}^{loclin}(x) &= \hat{\beta}_{0x} \quad \text{where} \\ \hat{\beta}_{0x}, \hat{\beta}_{1x} &= \text{argmin}_{\beta_{0x},\beta_{1x}} \sum_{i=1}^{n}\{Y_i - \beta_{0x} - \beta_{1x}(x_i - x)\}^2 K\Big(\frac{x - x_i}{h_n}\Big)\end{aligned}$$

- **Exercise 11.3** Suppose we define an estimator $\tilde{m}_{h_n}(x)$ of the regression function as

$$\begin{aligned}\tilde{m}_{h_n}(x) &= \hat{\beta}_{0x} \quad \text{where} \\ \hat{\beta}_{0x} &= \text{argmin}_{\beta_{0x}} \sum_{i=1}^{n}\{Y_i - \beta_{0x}\}^2 K\Big(\frac{x - x_i}{h_n}\Big)\end{aligned}$$

  Show that $\tilde{m}_{h_n}(x) = \hat{m}_{h_n}^{NW}(x)$.

11.3.2 Advantages of the Local Linear Estimator

- The local linear regression estimator can reduce the effects of design and boundary bias.
- If we write the local linear estimate at the point x as $\hat{m}_{h_n}^{loclin}(x) = \sum_{i=1}^{n} a_i^{h_n}(x) Y_i$, then the bias is appoximately

$$E\{\hat{m}_{h_n}^{loclin}(x)\} - m(x) \approx m'(x)\sum_{i=1}^{n}(x_i - x)a_i^{h_n}(x) + \frac{m''(x)}{2}\sum_{i=1}^{n}(x_i - x)^2 a_i^{h_n}(x)$$

- For local linear regression, the term $m'(x)\sum_{i=1}^{n}(x_i - x)a_i^{h_n}(x)$ equals zero. If the weights $a_i^{h_n}(x)$ were the weights from the Nadaraya-Watson estimator, this term would not necessarily equal zero.
- Also, the local linear estimator can help to reduce the boundary bias that arises from asymmetry near the boundary (draw a picture).

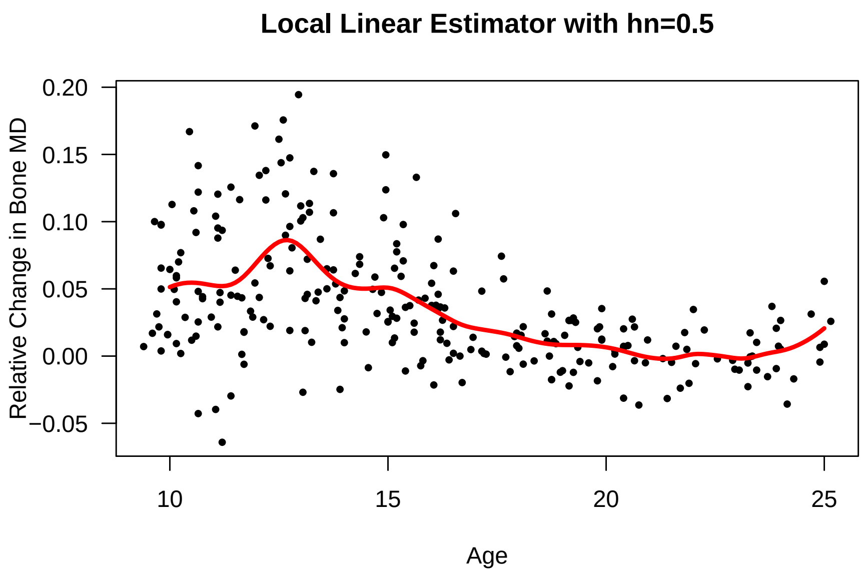
11.3.3 An Example in R

- An R function which implements local linear regression is the following. The input for this function has the same structure as our earlier Nadaraya-Watson R function.

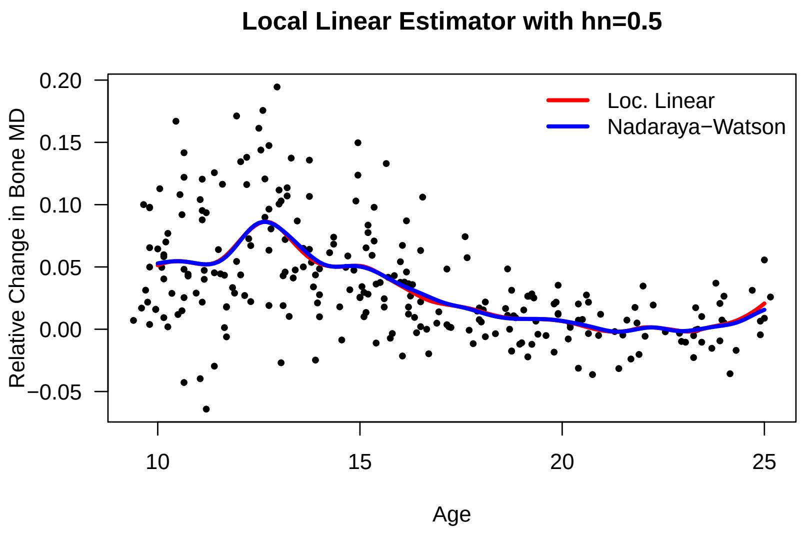
```
MyLocLinear <- function(x, y, bandwidth, x.points) {
  q <- length(x.points)
  loclin.est <- numeric(q)
  for(k in 1:q) {
    ## First create weights with Gaussian kernel
    xtmp <- x - x.points[k]
    ww <- dnorm(xtmp, mean=0, sd=bandwidth)

    ## Now, compute the intercept with a weighted linear regression
    loclin.est[k] <- lm(y ~ xtmp, weights=ww)$coef[1]
  }
  return(loclin.est)
}
```

- Let's try this function with the `bonedat` dataset again.
- Using age as the covariate, we will estimate the regression function at the points $10, 10.1, 10.2, ..., 25$:

```
xseq <- seq(10, 25, by=.1)
bone.loclin <- MyLocLinear(x=bonedat$age, y=bonedat$spnbmd,
                           bandwidth=0.5, x.points=xseq)

plot(bonedat$age, bonedat$spnbmd, las=1, ylab="Relative Change in Bone MD",
     xlab="Age", main="Local Linear Estimator with hn=0.5", type="n")
points(bonedat$age, bonedat$spnbmd, pch=16, cex=0.7)
lines(xseq, bone.loclin, lwd=3, col="red")
```



- Let's compare this with the Nadaraya-Watson esitmate that we computed



earlier

11.3.4 Local Polynomial Regression

- There is no reason why we must restrict ourselves to local linear fits. We could also fit local polynomial models.
- Similar to the way we approached local linear regression, for a fixed x we will fit the local model

$$\hat{s}_x^p(x_i) = \hat{\beta}_{0x,p} + \hat{\beta}_{1x,p}(x_i - x) + \hat{\beta}_{2x,p}(x_i - x)^2 + ... + \beta_{px,p}(x_i - x)^p,$$

  where the estimated regression coefficients $\hat{\beta}_{0x,p}, \hat{\beta}_{1x,p}, \dots, \hat{\beta}_{px,p}$ are found by solving the least-squares problem

$$\sum_{i=1}^{n}\{Y_i - \beta_{0x,p} - \beta_{1x,p}(x_i - x) - ... - \beta_{px,p}(x_i - x)^p\}^2 K\Big(\frac{x - x_i}{h_n}\Big) \quad (11.6)$$

- Then, we estimate the regression function at x with $\hat{m}_{h_n}^{locpoly}(x) = \hat{s}_x^p(x) = \hat{\beta}_{0x,p}$.
- Note that the local linear regression estimate is just a special case of local polynomial regression with $p = 1$.

- To find the estimates of $\beta_{0x,p}$ for linear and polynomial regression, you can use the formulas for the regression coefficient estimates in weighted least squares.

- Define the $n \times n$ diagonal matrix of weights $\mathbf{W}_{x,h_n}$ as

$$\mathbf{W}_{x,h_n} = \begin{bmatrix} K\left(\frac{x-x_1}{h_n}\right) & 0 & \dots & 0 \\ 0 & K\left(\frac{x-x_2}{h_n}\right) & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & K\left(\frac{x-x_n}{h_n}\right) \end{bmatrix}$$

and define the $n \times (p+1)$ matrix $\mathbf{X}_{x,p}$ as

$$\mathbf{X}_{x,p} = \begin{bmatrix} 1 & (x_1 - x) & \dots & (x_1 - x)^p \\ 1 & (x_2 - x) & \dots & (x_2 - x)^p \\ \vdots & \vdots & & \vdots \\ 1 & (x_n - x) & \dots & (x_n - x)^p \end{bmatrix}$$

- The vector of estimated regression coefficients is obtained from the following formula

$$\begin{bmatrix} \hat{\beta}_{0x,p} \\ \hat{\beta}_{1x,p} \\ \dots \\ \hat{\beta}_{px,p} \end{bmatrix} = (\mathbf{X}_{x,p}^T \mathbf{W}_{x,h_n} \mathbf{X}_{x,p})^{-1} \mathbf{X}_{x,p}^T \mathbf{W}_{x,h_n} \mathbf{Y}$$

- While using local polynomial regression with higher order polynomials offer more flexibility, they come at the price of more variance.

- For fixed h_n, increasing the degree p can decrease bias but will increase variance.

- In practice, $p = 1$ or $p = 2$ seems to be most common in practice. There is often not much benefit to using a degree of 3 or more.

11.4 Selecting the Bandwidth/Smoothing Parameter

11.4.1 Representing in Linear Form

- Let $\mathbf{Y} = (Y_1, \dots, Y_n)$ and let $\hat{\mathbf{m}} = (\hat{m}(x_1), \dots, \hat{m}(x_n))$ denote the vector of "fitted values" from a vector of estimates of the regression function at $x_1, \dots, x_n$.

- You can represent the fitted values all of the nonparametric estimators discussed thus far as
$$\hat{\mathbf{m}} = \mathbf{A}_{h_n}\mathbf{Y}$$
for an appropriately chosen $n \times n$ matrix $\mathbf{A}_{h_n}$.

- For the local average estimator, we have $\hat{\mathbf{m}} = \mathbf{A}_{h_n}\mathbf{Y}$ where $\mathbf{A}_{h_n}$ is defined as

$$\mathbf{A}_{h_n} = \begin{bmatrix} \frac{1}{n_{h_n}(x_1)} I(x_1 - h_n < x_1 < x_1 + h_n) & \dots & \frac{1}{n_{h_n}(x_1)} I(x_1 - h_n < x_n < x_1 + h_n) \\ \frac{1}{n_{h_n}(x_2)} I(x_2 - h_n < x_1 < x_2 + h_n) & \dots & \frac{1}{n_{h_n}(x_2)} I(x_2 - h_n < x_n < x_2 + h_n) \\ \vdots & \ddots & \vdots \\ \frac{1}{n_{h_n}(x_n)} I(x_n - h_n < x_1 < x_n + h_n) & \dots & \frac{1}{n_{h_n}(x_n)} I(x_n - h_n < x_n < x_n + h_n) \end{bmatrix} \tag{11.7}$$

where $n_{h_n}(x) = \sum_{i=1}^{n} I(x - h_n < x_i < x + h_n)$.

- In other words, the (i, j) element of $\mathbf{A}_{h_n}$ is $a_j(x_i)$ where

$$a_j(x_i) = \frac{1}{n_{h_n}(x_i)} I(x_i - h_n < x_j < x_i + h_n)$$

- For the Nadaraya-Watson estimator, the $\mathbf{A}_{h_n}$ matrix is

$$\mathbf{A}_{h_n} = \begin{bmatrix} \frac{1}{K_{h_n}(x_1,\cdot)} K(0) & \frac{1}{K_{h_n}(x_1,\cdot)} K(\frac{x_2 - x_1}{h_n}) & \dots & \frac{1}{K_{h_n}(x_1,\cdot)} K(\frac{x_n - x_1}{h_n}) \\ \frac{1}{K_{h_n}(x_2,\cdot)} K(\frac{x_1 - x_2}{h_n}) & \frac{1}{K_{h_n}(x_2,\cdot)} K(0) & \dots & \frac{1}{K_{h_n}(x_2,\cdot)} K(\frac{x_n - x_2}{h_n}) \\ \vdots & \vdots & \ddots & \vdots \\ \frac{1}{K_{h_n}(x_n,\cdot)} K(\frac{x_1 - x_n}{h_n}) & \frac{1}{K_{h_n}(x_n,\cdot)} K(\frac{x_2 - x_n}{h_n}) & \dots & \frac{1}{K_{h_n}(x_n,\cdot)} K(0) \end{bmatrix}$$

where
$$K_{h_n}(x_i, \cdot) = \sum_{j=1}^{n} K\Big(\frac{x_i - x_j}{h_n}\Big)$$

- For the local linear regression estimator, the i^{th} row of $\mathbf{A}_{h_n}$ equals the first row of the following $2 \times n$ matrix:

$$(\mathbf{X}_{x_i,1}^T \mathbf{W}_{x_i,h_n} \mathbf{X}_{x_i,1})^{-1} \mathbf{X}_{x_i,1}^T \mathbf{W}_{x_i,h_n}$$

Here, $\mathbf{X}_{x,1}$ and $\mathbf{W}_{x,h_n}$ are as defined in the section on local linear regression.

- In “classic” linear regression where you would try to fit the straight line model $Y_i = \beta_0 + \beta_1 x_i + \varepsilon_i$, the vector of fitted values would be

$$\hat{\mathbf{m}} = (\hat{m}(x_1), \dots, \hat{m}(x_n)) = (\hat{\beta}_0 + \hat{\beta}_1 x_1, \dots, \hat{\beta}_0 + \hat{\beta}_1 x_n)$$

- In this case, you can represent $\hat{\mathbf{m}}$ as

$$\hat{\mathbf{m}} = \mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y}$$

where $\mathbf{X}$ is the following $n \times 2$ “design” matrix

$$\mathbf{X} = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_n \end{bmatrix}$$

11.4.2 The Cp Statistic

- **Theorem:** If a random vector $\mathbf{Z}$ has mean vector   and covariance matrix , then

$$E\{\mathbf{Z}^T\mathbf{Z}\} = E\Big\{\sum_{i=1}^n Z_i^2\Big\} = {}^{T} \; + \mathrm{tr}(\;)$$

- Notice that the vector $\mathbf{m} - \mathbf{A}_{h_n}\mathbf{Y}$ has

$$E(\mathbf{m} - \mathbf{A}_{h_n}\mathbf{Y}) = (\mathbf{I} - \mathbf{A}_{h_n})\mathbf{m} \qquad \mathrm{Var}(\mathbf{m} - \mathbf{A}_{h_n}\mathbf{Y}) = \sigma^2 \mathbf{A}_{h_n}\mathbf{A}_{h_n}^T$$

- Also, the vector $\mathbf{Y} - \mathbf{A}_{h_n}\mathbf{Y} = (\mathbf{I} - \mathbf{A}_{h_n})\mathbf{Y}$ has

$$E\{(\mathbf{I}-\mathbf{A}_{h_n})\mathbf{Y}\} = (\mathbf{I}-\mathbf{A}_{h_n})\mathbf{m} \qquad \mathrm{Var}\{(\mathbf{I}-\mathbf{A}_{h_n})\mathbf{Y}) = \sigma^2(\mathbf{I}-\mathbf{A}_{h_n})(\mathbf{I}-\mathbf{A}_{h_n})^T$$

- Ideally, we would like to choose the smoothing parameter h_n to minimize the following mean averaged squared error

$$\begin{aligned} \mathrm{MASE}(h_n) &= \frac{1}{n}\sum_{i=1}^n E\Big[\{m(x_i) - \hat{m}(x_i)\}^2\Big] \\ &= E\{\frac{1}{n}[\mathbf{m} - \mathbf{A}_{h_n}\mathbf{Y}]^T[\mathbf{m} - \mathbf{A}_{h_n}\mathbf{Y}]\} \end{aligned}$$

- If we apply the above Theorem to the vector $\mathbf{m} - \mathbf{A}_{h_n}\mathbf{Y}$, we can notice that

$$\begin{aligned}\text{MASE}(h_n) &= E\{\frac{1}{n}(\mathbf{m} - \mathbf{A}_{h_n}\mathbf{Y})^T(\mathbf{m} - \mathbf{A}_{h_n}\mathbf{Y})\} \\ &= \frac{1}{n}[(\mathbf{I} - \mathbf{A}_{h_n})\mathbf{m}]^T[(\mathbf{I} - \mathbf{A}_{h_n})\mathbf{m}] + \frac{\sigma^2}{n}\text{tr}(\mathbf{A}_{h_n}\mathbf{A}_{h_n}^T) \qquad (11.8)\end{aligned}$$

- Now, using the mean and covariance matrix for $(\mathbf{I} - \mathbf{A}_{h_n})\mathbf{Y}$, we also have that

$$\begin{aligned}&E\{\frac{1}{n}[(\mathbf{I} - \mathbf{A}_{h_n})\mathbf{Y}]^T[(\mathbf{I} - \mathbf{A}_{h_n})\mathbf{Y}]\} \\ =\ & \frac{1}{n}[(\mathbf{I} - \mathbf{A}_{h_n})\mathbf{m}]^T[(\mathbf{I} - \mathbf{A}_{h_n})\mathbf{m}] + \frac{\sigma^2}{n}\text{tr}\{(\mathbf{I} - \mathbf{A}_{h_n})(\mathbf{I} - \mathbf{A}_{h_n})^T\} \\ =\ & \text{MASE}(h_n) + \sigma^2 - \frac{2\sigma^2}{n}\text{tr}(\mathbf{A}_{h_n}) \qquad (11.9)\end{aligned}$$

- So, if σ^2 is known, then $\widehat{\text{MASE}}(h_n)$ is an unbiased estimate of $\text{MASE}(h_n)$

$$\begin{aligned}\widehat{\text{MASE}}(h_n) &= \frac{1}{n}[(\mathbf{I} - \mathbf{A}_{h_n})\mathbf{Y}]^T[(\mathbf{I} - \mathbf{A}_{h_n})\mathbf{Y}] - \sigma^2 + \frac{2\sigma^2}{n}\text{tr}(\mathbf{A}_{h_n}) \\ &= \frac{1}{n}\sum_{i=1}^{n}\{Y_i - \hat{m}_{h_n}(x_i)\}^2 - \sigma^2 + \frac{2\sigma^2}{n}\text{tr}(\mathbf{A}_{h_n})\end{aligned}$$

- The predictive mean averaged squared error (PMASE) is defined as

$$\text{PMASE}(h_n) = E\Big[\frac{1}{n}\sum_{i=1}^{n}\{Y_i' - \hat{m}_{h_n}(x_i)\}^2\Big] \qquad (11.10)$$

  where Y_i' is a "future" independent observation that has the same covariate as Y_i.

- We assume that $Y_i' = m(x_i) + \varepsilon_i'$ where ε_i' is independent of ε_i.

- So,

$$\begin{aligned}\text{PMASE}(h_n) &= E\Big[\frac{1}{n}\sum_{i=1}^{n}\{m(x_i) - \hat{m}_{h_n}(x_i) + \varepsilon_i'\}^2\Big] \\ &= E\Big[\frac{1}{n}\sum_{i=1}^{n}\{m(x_i) - \hat{m}_{h_n}(x_i)\}^2\Big] + E\Big[\frac{1}{n}\sum_{i=1}^{n}(\varepsilon_i')^2\Big] \\ &= \text{MASE}(h_n) + \sigma^2\end{aligned}$$

- The C_p statistic is based on the idea that, if σ^2 was known, then the following quantity would be an unbiased estimate of PMASE(h_n):

$$\frac{1}{n}\sum_{i=1}^{n}\{Y_i - \hat{m}_{h_n}(x_i)\}^2 + \frac{2\sigma^2}{n}\text{tr}(\mathbf{A}_{h_n})$$

- The C_p statistic formula is obtained by plugging in an estimate $\hat{\sigma}^2$ of the residual variance into the above formula:

$$C_p(h_n) = \frac{1}{n}\sum_{i=1}^{n}\{Y_i - \hat{m}_{h_n}(x_i)\}^2 + \frac{2\hat{\sigma}^2}{n}\text{tr}(\mathbf{A}_{h_n})$$

- The reason this is called the "C_p statistic" is that in the case of a linear regression model with p columns in the design matrix, we have $\hat{\mathbf{m}} = \mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y}$ and $\text{tr}\{\mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\} = p$ so an estimator of the PMASE would be

$$C_p = \frac{1}{n}\sum_{i=1}^{n}\{Y_i - \hat{m}_{h_n}(x_i)\}^2 + \frac{2\hat{\sigma}^2}{n}p$$

- For this reason, $\text{tr}(\mathbf{A}_{h_n})$ is often referred to as the "degrees of freedom" of a nonparametric estimator of the form $\hat{\mathbf{m}} = \mathbf{A}_{h_n}\mathbf{Y}$.

- The main drawback of the C_p statistic is that it requires an estimate of the residual variance σ^2.
- So, if we choose a fairly small bandwidth $\tilde{h}_n$ so that the bias is close to zero, we could use the following estimate of σ^2

$$\hat{\sigma}^2(\tilde{h}_n) = \frac{\sum_{i=1}^{n}\{Y_i - \hat{m}_{\tilde{h}_n}(x_i)\}^2}{n - 2\text{tr}(\mathbf{A}_{\tilde{h}_n}) + \text{tr}(\mathbf{A}_{\tilde{h}_n}\mathbf{A}_{\tilde{h}_n}^T)}$$

- **Exercise 11.4** Suppose the $n \times n$ matrix $\mathbf{A}_{h_n}$ satifies $\mathbf{A}_{h_n}\mathbf{m} = \mathbf{m}$. Show that

$$\frac{\mathbf{Y}^T(\mathbf{I} - \mathbf{A}_{h_n})^T(\mathbf{I} - \mathbf{A}_{h_n})\mathbf{Y}}{n - 2\text{tr}(\mathbf{A}_{h_n}) + \text{tr}(\mathbf{A}_{h_n}\mathbf{A}_{h_n}^T)}$$

is an unbiased estimator of σ^2.

11.4.3 Leave-one-out Cross Validation

- Similar to the way we defined a leave-on-out density estimate in Chapter 8, we will define the leave-one-out estimate of the regression function at x as:

$$\hat{m}_{h_n,-i}(x) - \text{ estimate of } m(x) \text{ found by using all data except } (Y_i, x_i). \tag{11.11}$$

- The leave-one-out cross validation (LOO-CV) estimate of the PMASE is defined to be

$$\text{LOOCV}(h_n) = \frac{1}{n}\sum_{i=1}^{n}\{Y_i - \hat{m}_{h_n,-i}(x_i)\}^2$$

- The intuition here is that; because the estimate $\hat{m}_{h_n,-i}(x_i)$ is computed without the i^{th} observation, Y_i plays the role of a "future observation" (relative to the dataset that does not contain Y_i).
- Hence, $\{Y_i - \hat{m}_{h_n,-i}(x_i)\}^2$ should be a sensible replacement for the unobservable $\{Y_i' - \hat{m}_{h_n}(x_i)\}^2$.

- While we could compute $\text{LOOCV}(h_n)$ by computing $\hat{m}_{h_n,-i}(x_i)$ separately for $i = 1, \dots, n$, there is a much more efficient way of computing $\text{LOOCV}(h_n)$.
- We are assuming that $\hat{m}_{h_n}(x)$ can be represented as a linear combination of the responses

$$\hat{m}_{h_n}(x) = \sum_{j=1}^{n} a_j^{h_n}(x) Y_j,$$

  where we can think of $a_j^{h_n}(x)$ as weights that sum to 1.
- If we did not use Y_i to compute $\hat{m}_{h_n}(x)$, this estimate would look like

$$\hat{m}_{h_n,-i}(x) = \sum_{j=1}^{n} a_{j,-i}^{h_n}(x) Y_j \tag{11.12}$$

  where

$$a_{j,-i}^{h_n}(x) = \begin{cases} 0 & \text{if } j = i \\ \frac{a_j^{h_n}(x)}{\sum_{k \neq i} a_k^{h_n}(x)} & \end{cases}$$

- If you want to better convince yourself that the above formula for $a_{j,-i}^{h_n}(x_i)$ is true, try an example using the Nadaraya-Watson estimator with $n = 3$.

- Because $\sum_{k \neq i} a_k^{h_n}(x_i) = 1 - a_i^{h_n}(x_i)$, we can express $Y_i - \hat{m}_{h_n,-i}(x_i)$ as

$$\begin{aligned} Y_i - \hat{m}_{h_n,-i}(x_i) &= Y_i - \sum_{j \neq i}^{n} a_{j,-i}^{h_n}(x_i) Y_j \\ &= Y_i - \frac{1}{1 - a_i^{h_n}(x_i)} \sum_{j \neq i}^{n} a_j^{h_n}(x_i) Y_j \\ &= Y_i - \Big[\frac{1}{1 - a_i^{h_n}(x_i)} \sum_{j=1}^{n} a_j^{h_n}(x_i) Y_j \Big] + \frac{a_j^{h_n}(x_i) Y_i}{1 - a_i^{h_n}(x_i)} \\ &= \frac{Y_i - \hat{m}_{h_n}(x_i)}{1 - a_i^{h_n}(x_i)} \end{aligned} \tag{11.13}$$

- Using (11.13), we can re-write the LOOCV estimate as

$$\text{LOOCV}(h_n) = \frac{1}{n} \sum_{i=1}^{n} \Big(\frac{Y_i - \hat{m}_{h_n}(x_i)}{1 - a_i^{h_n}(x_i)} \Big)^2$$

where $a_i^{h_n}(x_i)$ are just the diagonal elements of our original matrix $\mathbf{A}_{h_n}$.

11.4.4 Example: Choosing the Best Bin Width for the Local Average Estimator.

Cp Statistic

- The diagonal entries of the $\mathbf{A}_{h_n}$ matrix for the local average estimator are $1/n_{h_n}(x_1), \ldots, 1/n_{h_n}(x_n)$.
- So, the "degrees of freedom" for the local average estimator is

$$\text{tr}(\mathbf{A}_{h_n}) = \sum_{i=1}^{n} \frac{1}{n_{h_n}(x_i)} \tag{11.14}$$

- Notice that if we choose h_n large enough so that $n_{h_n}(x_i) = n$ for all x_i, then the degrees of freedom is equal to 1.
- The C_p statistic for the local average estimator is

$$\frac{1}{n} \sum_{i=1}^{n} \{Y_i - \hat{m}_{h_n}(x_i)\}^2 + \frac{2\hat{\sigma}^2}{n} \sum_{i=1}^{n} \frac{1}{n_{h_n}(x_i)}$$

- Let's try to compute $C_p(h_n)$ for the `bonedat` dataset.
- The first step is to write a function that computes the $n_{h_n}(x_i)$ for a given value of h_n. This will allow us to find the degrees of freedom and will also be helpful later when computing LOOCV.

```
NumInBins <- function(hh, xx) {
  ## This function returns a vector of length n
  ## Elements of this vector will be: n_[h_n](x_1), n_[h_n](x_2), ...
  n <- length(xx)
  num.bin <- numeric(n)
  for(k in 1:n) {
    num.bin[k] <- sum(xx > xx[k] - hh & xx < xx[k] + hh)
  }
  return(num.bin)
}
```

- We also want a function that returns the vector with elements $\hat{m}_{h_n}(x_1), \hat{m}_{h_n}(x_2), ... \hat{m}_{h_n}(x_n)$.

```
MyLocAvgEst <- function(xx, yy, hh) {
  n <- length(xx)
  m.hat.loc <- numeric(n)
  for(k in 1:n) {
    in.bin <- xx > xx[k] - hh & xx < xx[k] + hh
    m.hat.loc[k] <- mean(yy[in.bin])
  }
  return(m.hat.loc)
}
```

- The final step is to compute an estimate of σ^2.
- Using the estimate

$$\hat{\sigma}^2(\tilde{h}_n) = \frac{\sum_{i=1}^n \{Y_i - \hat{m}_{\tilde{h}_n}(x_i)\}^2}{n - 2\mathrm{tr}(\mathbf{A}_{\tilde{h}_n}) + \mathrm{tr}(\mathbf{A}_{\tilde{h}_n}\mathbf{A}_{\tilde{h}_n}^T)} \tag{11.15}$$

that we mentioned before with $\tilde{h}_n = 0.1$, I got a an estimate of σ^2 which was quite close to 0.0015

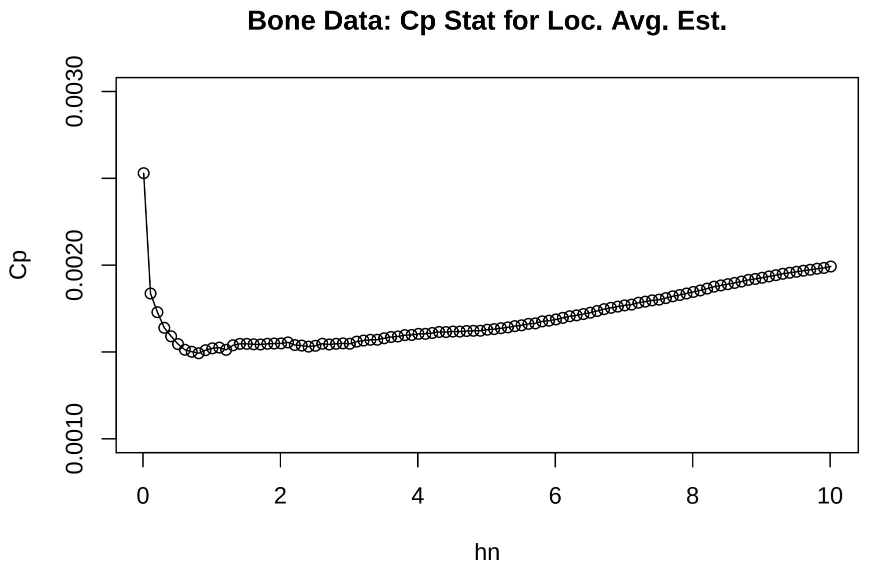
```r
sigsq.est <- 0.0015
```

- Now, we are ready to compute the C_p statistic. We will compute $C_p(h_n)$ for $h_n = 0.01, 0.11, \ldots, 10.01$. This can be done with the following code:

```r
hseq <- seq(.01, 10.01, by=.1)
ngrid <- length(hseq)
n <- length(bonedat$age)
Cp <- numeric(ngrid)
for(k in 1:ngrid) {
   m.hat <- MyLocAvgEst(bonedat$age, bonedat$spnbmd, hseq[k])
   dfval <- sum(1/NumInBins(hseq[k], bonedat$age))
   Cp[k] <- mean( (bonedat$spnbmd - m.hat)^2 ) + (2*sigsq.est*dfval)/n
}
```

- We can plot the values of $C_p(h_n)$ vs. h_n to roughly see where the minimum value is. From the graph, it looks to be slighly less than 1.

```r
plot(hseq, Cp, ylim=c(0.001,.003), main="Bone Data: Cp Stat for Loc. Avg. Est.",
     xlab="hn", ylab="Cp")
lines(hseq, Cp)
```



- More precisely, the value of h_n from our sequence which has the smallest value of $C_p(h_n)$ is 0.81.

```
hseq[which.min(Cp)]
```

```
## [1] 0.81
```

LOOCV

- We can use the functions that we have written to compute LOOCV(h_n).

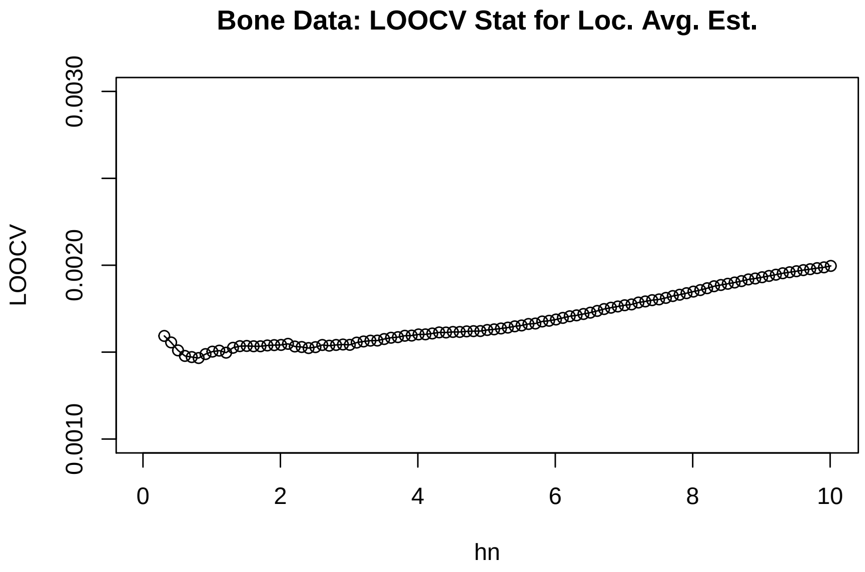
- It is useful to notice that $1 - a_i^{h_n}(x_i) = 1 - 1/n_{h_n}(x_i)$ using the notation we used in the description of the LOOCV.

- `R` code to compute LOOCV(h_n) at the same sequence of h_n values used for the C_p statistic is given below:

```
LOOCV <- numeric(ngrid)
for(k in 1:ngrid) {
  m.hat <- MyLocAvgEst(bonedat$age, bonedat$spnbmd, hseq[k])
  n.hn <- NumInBins(hseq[k], bonedat$age)
  dd <- 1 - 1/n.hn
  LOOCV[k] <- mean( ((bonedat$spnbmd - m.hat)/dd)^2 )
}
```

- We can plot the values of LOOCV(h_n) vs. h_n to roughly see where the minimum value is.

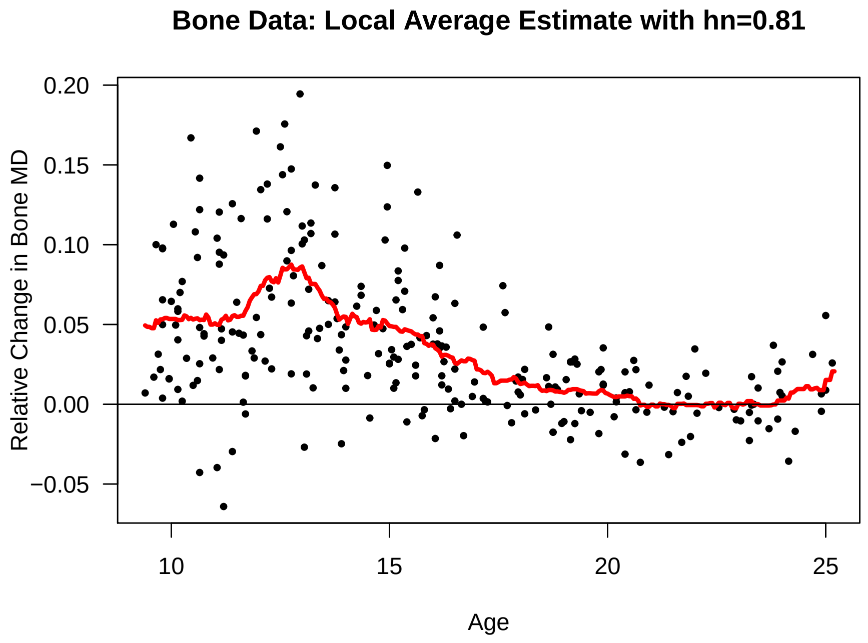
```
plot(hseq, LOOCV, ylim=c(0.001,.003), main="Bone Data: LOOCV Stat for Loc. Avg. Est.",
     xlab="hn", ylab="LOOCV")
lines(hseq, LOOCV)
```



- The value of h_n from our sequence which has the smallest value of LOOCV(h_n) is 0.81.

```r
hseq[which.min(LOOCV)]
```

```
## [1] 0.81
```



11.5 Additional functions in R

- The R functions `lowess` and `loess` are widely used functions for smoothing via local regression.
- `loess` is basically an expanded version of `lowess`. The function `loess` has more options than the `lowess` function and is written to resemble the `lm` in function in `R`.
- Note that `loess` and `lowess` have different default settings for some of the model fitting parameters so they can differ somewhat in the values they return unless you set these parameters to be equal.
- `loess` allows for multivariate covariates $\mathbf{x}_i$ while `lowess` does not.

- `lowess` does local quadratic and local linear regression. The format of the `lowess` function is the following:

```
loess(formula, data, span)
```

- **formula** - usally of the form `y ~ x` if using a single response vector `y` and covariate `x`
- **data** - the dataset from which `y` and `x` are drawn from
- **span** - the "span" of the smoother. This can be thought of as playing the role of the bandwidth. Default of span $= \alpha$ is set to 0.75. Usually, $0 < \alpha < 1$.
- **degree** - the degree of the polynomial used for local regression. The default is set to 2.
- The `loess` function will return two vectors `$x` and `$fitted`. The `$x` is just the vector of covariates from the original data, and the `$fitted` vector is the value of the estimated regression function $\hat{m}(x_i)$ at these points.
- The `lowess` function performs local linear regression with a few extra steps included to make the fitting procedure more robust.

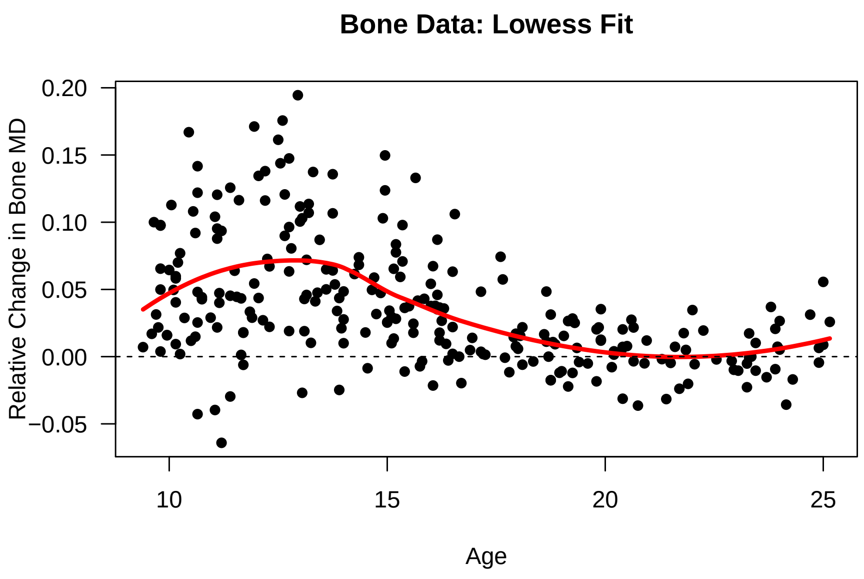
- For weights $W_i^{\alpha}(x)$ in the local linear regression problem, loess and lowess uses the following tri-cube function

$$W_i^{\alpha}(x) = \begin{cases} \left(1 - \left(\frac{|x-x_i|}{|x-x_{(q)}|}\right)^3\right)^3 & \text{if } |x - x_i| \leq |x - x_{(q)}| \\ 0 & \text{otherwise} \end{cases} \tag{11.16}$$

- The weights $W_i^{\alpha}(x)$ here play the same role as $K(\frac{x-x_i}{h_n})$ did in our description of local linear regression in Section 10.3.
- Here, $x_{(q)}$ is the $\lfloor \alpha \times n \rfloor$ furthest observations away from x, and $\lfloor \alpha \times n \rfloor$ denotes αn rounded down to the nearest integer.
- So, values of the span α which are closer to 0 mean that you are using a smaller number of observations when performing each local regression while values close to 1 mean that nearly all the observations receive positive weight when performing each local regression.
- After fitting a local regression with weights $W_i^{\alpha}(x)$, `lowess` actually does an additional local
  regression with updated weights that reduce the influence of outliers. `loess` will do the same thing if `family` is set to "symmetric" rather than "gaussian".

- Let's try plotting a `loess` fit using the bone data. We will set `span = 2/3` instead of 3/4

```
bone.low.fit <- loess(spnbmd ~ age, data=bonedat, span=2/3)

plot(bone.low.fit, ylab="Relative Change in Bone MD",
     xlab="Age", main="Bone Data: Lowess Fit", las=1, pch=16)
## plot a line of the the fitted values vs. x. Need to sort
## the x_i's first though if we want a nice looking line:
lines(bone.low.fit$x[order(bone.low.fit$x)], bone.low.fit$fitted[order(bone.low.fit$x)]
      col="red", lwd=3)
abline(0,0, lty=2)
```



- If you want to change the settings for `lowess` and `loess` so that they are using the exact same fitting procedure, you can use the following approach:

```
lowess.fit <- lowess(x=bonedat$age, y=bonedat$spnbmd, iter=3, delta=0, f=2/3)
loess.fit <- loess(spnbmd ~ age, data=bonedat, span=2/3, degree=1, family="symmetric",
                   iterations=4, surface="direct")
```

- The `locpoly` function from the `KernSmooth` package implements local polynomial regression as it was described in Section 10.3.

```
locpoly(x, y, degree, kernel = "normal", bandwidth)
```

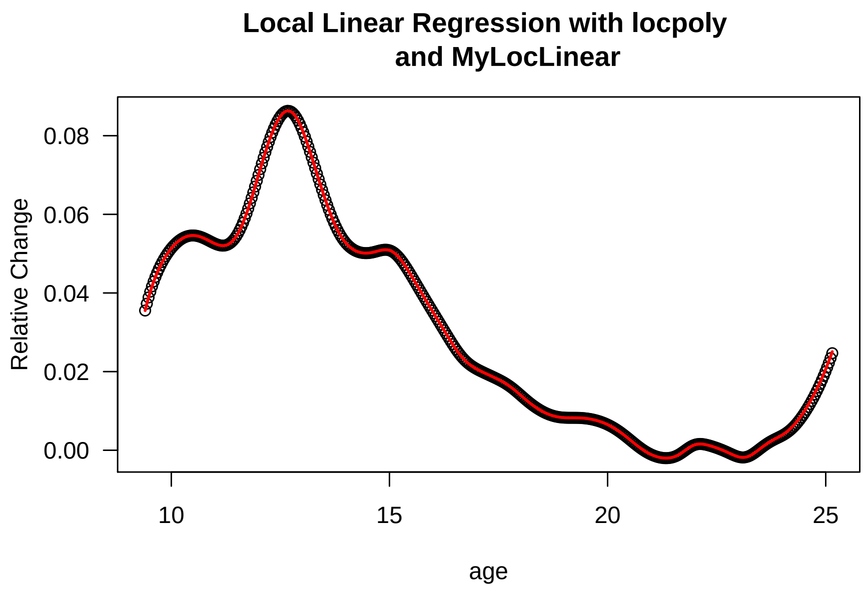
- You can vary the degree of the polynomical used in the local polynomial regression with the `degree` argument.

- For local linear regression, you should get the same answer as our function `MyLocLinear` written in Section 10.3 if you use `degree=1` with the locpoly function:

```
library(KernSmooth)
```

```
## KernSmooth 2.23 loaded
## Copyright M. P. Wand 1997-2009
```

```
locpoly.fit <- locpoly(x = bonedat$age, y=bonedat$spnbmd, degree=1, bandwidth=0.5)
mylocpoly.fit <- MyLocLinear(x=bonedat$age, y=bonedat$spnbmd, bandwidth=0.5,
                             x.points=locpoly.fit$x)


plot(locpoly.fit$x, locpoly.fit$y, main="Local Linear Regression with locpoly
     and MyLocLinear", las=1, xlab="age", ylab="Relative Change")
lines(locpoly.fit$x, mylocpoly.fit, col="red", lwd=2)
```



- The `supsmu` function implements Friedman's "super smoother" (Friedman (1984)).

```r
supsmu(x, y, span="cv", bass=0)
```

- **x** - vector of covariate values.
- **y** - vector of responses.
- **span** - the "span" of the smoother. As with loess, this is the fraction of observations included when estimating the regression function at a particular point. The best span is chosen through cross-validation.
- **bass** - tuning parameter to further control the smoothness of the fitted curve. The smallest value is 0 (lowest level of smoothness). The largest value is 10 (most smoothness).

The `supsmu` function just returns a list with components `$x` and `$y`. The `$x` component will be a sorted vector of the inputted x values and the `$y` component will contain the corresponding estimates of the regression function.

11.6 Multivariate Problems

- All of the methods discussed here can be directly extended to the case of multivariate covariates $\mathbf{x}_i \in \mathbb{R}^p$ where $\mathbf{x}_i = (x_{i1}, \dots, x_{ip})$.
- For the methods that use a kernel function, we just use the Euclidean distance between two points in the kernel function.
- For example, with the Gaussian kernel for the Nadaraya-Watson estimator, we would use the following weights when estimating the regression function at $\mathbf{x} \in \mathbb{R}^p$

$$K\Big(\frac{\mathbf{x} - \mathbf{x}_i}{h_n}\Big) = \frac{1}{\sqrt{2\pi}h_n}\exp\Big\{-\frac{||\mathbf{x} - \mathbf{x}_i||^2}{2h_n^2}\Big\} = \frac{1}{\sqrt{2\pi}h_n}\exp\Big\{-\frac{1}{2h_n^2}\sum_{j=1}^{p}(x_j - x_{ij})^2\Big\}$$

- If we wanted to do k-nearest neighbors regression to estimate the regression function at $\mathbf{x}$, we would just pick the k closest observation in terms of the distance $||\mathbf{x} - \mathbf{x}_i||^2$.
- Similarly, if we wanted to compute the local average estimator in $\mathbf{R}^p$, we would use

$$\hat{m}_{h_n}^{loc}(\mathbf{x}) = \frac{1}{n_{h_n}(\mathbf{x})}\sum_{i=1}^{n} Y_i I(||\mathbf{x}_i - \mathbf{x}|| < h_n),$$

  where $n_{h_n}(\mathbf{x}) = \sum_{i=1}^{n} I(||\mathbf{x}_i - \mathbf{x}|| < h_n)$.

- The performance of kernel and local regression methods can degrade quickly as we move to higher dimensions. The convergence rate of the estimated regression function to the true regression function slows substantially as we increase p.
- "Curse of dimensionality" - need very large datasets to have a sufficient number of observations near a given point $\mathbf{x}$.
- The methods discussed here are most commonly used for $p = 1$ or maybe $p = 2$ or $p = 3$.

- For higher dimensions, modifications of these methods could be useful. For example:
  - Incorporating correlation between the covariates in the kernel function.

- Modeling the regression function as an additive model $m(\mathbf{x}) = m_1(x_1) + ... + m_j(x_p)$ and using local regression for each function $m_j(x)$.
- "Sparsity": try to first discard unimportant covariates and estimate the regression function using a smaller subset of the covariates.

11.7 Additional Reading

- Additional reading which covers the material discussed in this chapter includes:
  - Chapter 4 from Härdle et al. (2012)
  - Chapter 5 from Wasserman (2006)

Chapter 12

Splines and Penalized Regression

12.1 Introduction

- In this chapter, we will focus on using basis functions for estimating the regression function.
- That is, we will look at regression function estimates of the form

$$\hat{m}(x) = \hat{\beta}_0 \varphi_0(x) + \sum_{j=1}^{p} \hat{\beta}_j \varphi_j(x)$$

  where $\varphi_0(x), \varphi_1(x), \dots, \varphi_p(x)$ will be referred to as basis functions.

- We will usually either ignore $\varphi_0(x)$ or assume that $\varphi_0(x) = 1$.
- If you use a relatively large number of appropriately chosen basis functions, you can represent even quite complicated functions with some linear combination of the basis functions.

Examples

- Basis functions for a straight-line linear regression

$$\varphi_0(x) = 1 \qquad \varphi_1(x) = x$$

- Basis functions for a polynomial regression with degree 3

$$\varphi_0(x) = 1 \qquad \varphi_1(x) = x \qquad \varphi_2(x) = x^2 \qquad \varphi_3(x) = x^3$$

- Basis functions for a regressogram with bins $[l_k, u_k)$, $k = 1, \dots, K$:

$$\varphi_k(x) = I(l_k \leq x < u_k), \quad k = 1, \dots, K$$

12.1.1 Regressogram (Piecewise Constant Estimate)

- Let's consider the regressogram again. The regressogram estimate could be written as

$$\hat{m}_{h_n}^{R}(x) = \sum_{k=1}^{D_n} a_{k,h_n}\varphi_k(x) = \sum_{k=1}^{D_n} a_{k,h_n} I(x \in B_k) \tag{12.1}$$

where the coefficents a_{k,h_n} are given by

$$a_{k,h_n} = \frac{1}{n_{k,h_n}} \sum_{i=1}^{n} Y_i I(x_i \in B_k)$$

- We can think of the regressogram as a basis function estimate with the basis functions

$$\varphi_1(x) = I(x \in B_1), \dots, \varphi_{D_n}(x) = I(x \in B_{D_n})$$

- The regressogram estimate will be a piecewise constant function that is constant within each of the bins.

![Basis functions for a regressogram with the following 3 bins: [0,1/3), [1/3, 2/3), 2/3, 1)

12.1.2 Piecewise Linear Estimates

- Instead of a piecewise constant estimate of $m(x)$, we could use an estimate which is piecewise linear by using the following $2p$ basis functions

$$\begin{aligned}
\varphi_1(x) &= I(x \in B_1) \\
&\vdots \\
\varphi_p(x) &= I(x \in B_p) \\
\varphi_{p+1}(x) &= xI(x \in B_1) \\
&\vdots \\
\varphi_{2p}(x) &= xI(x \in B_p)
\end{aligned}$$

if we now let $p = D_n$ denote the number of "bins".

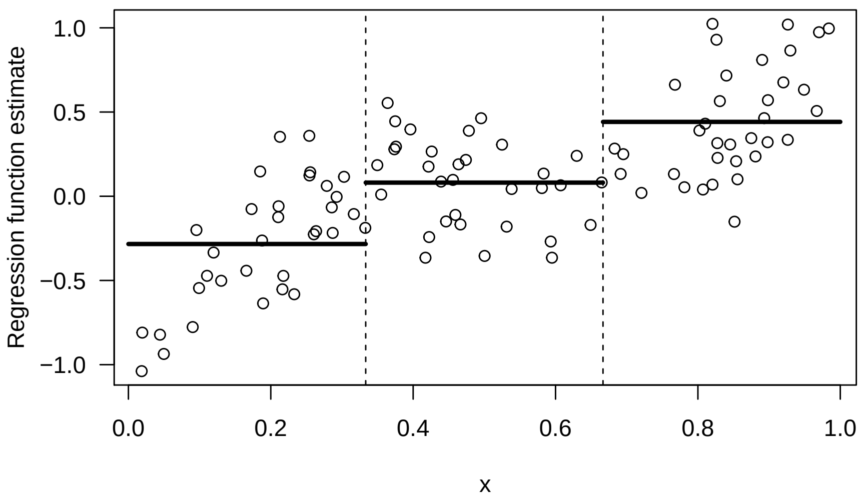


Figure 12.1: Regressogram estimate of a regression function with 3 bins.

- The figure below shows an example of a regression function that is piecewise linear with 3 different bins.

- While a piecewise linear model is perhaps a more flexible method than the regressogram, the piecwise linear model will still have big jumps at the bin boundaries and have an overall unpleasant appearance.

12.1.3 Piecewise Cubic Estimates

- If we wanted to allow for more flexible forms of the regression function estimate within each bin we could fit a higher order polynomial model within each bin.

- That is, the regression function estimate within the k^{th} bin will have the form
$$\hat{m}(x)I(x \in B_k) = \hat{\beta}_{0k} + \hat{\beta}_{1k}x + \hat{\beta}_{2k}x^2 + \hat{\beta}_{3k}x^3$$

- To fit a piecewise cubic model with p bins, we would need the following

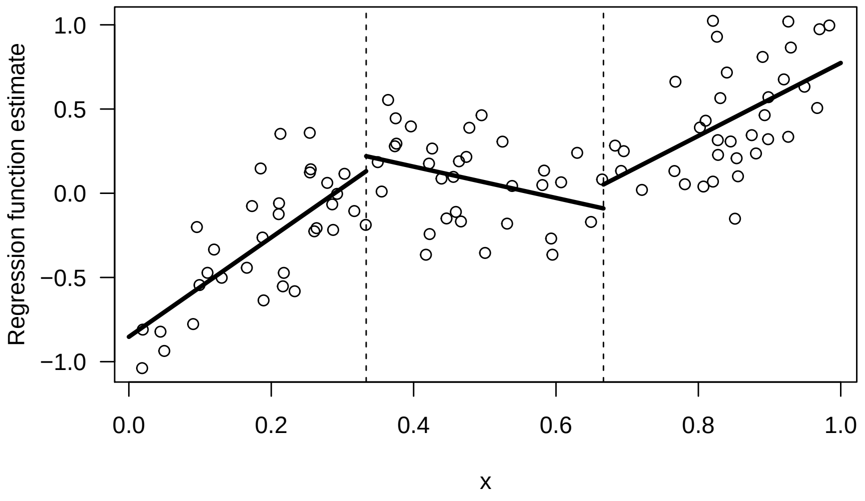


Figure 12.2: Example of a regression function estimate that is piecewise linear within 3 bins.

$4p$ basis functions

$$
\begin{aligned}
\varphi_1(x) &= I(x \in B_1), \ldots, \varphi_p(x) = I(x \in B_p) \\
\varphi_{p+1}(x) &= xI(x \in B_1), \ldots, \varphi_{2p}(x) = xI(x \in B_p) \\
\varphi_{2p+1}(x) &= x^2 I(x \in B_1), \ldots, \varphi_{3p}(x) = x^2 I(x \in B_p) \\
\varphi_{3p+1}(x) &= x^3 I(x \in B_1), \ldots, \varphi_{4p}(x) = x^3 I(x \in B_p)
\end{aligned}
\tag{12.2}
$$

12.2 Piecewise Linear Estimates with Continuity (Linear Splines)

- In the spline world, one typically talks about “knots” rather than “bins”.
- You can think of knots as the dividing points between the bins.
- We will let $u_1 < u_2 < \ldots < u_q$ denote the choice of knots.
- The bins corresponding to this set of knots would then be $B_1 = (-\infty, u_1), B_2 = [u_1, u_2), B_3 = [u_2, u_3), \ldots, B_{q+1} = [u_q, \infty)$.

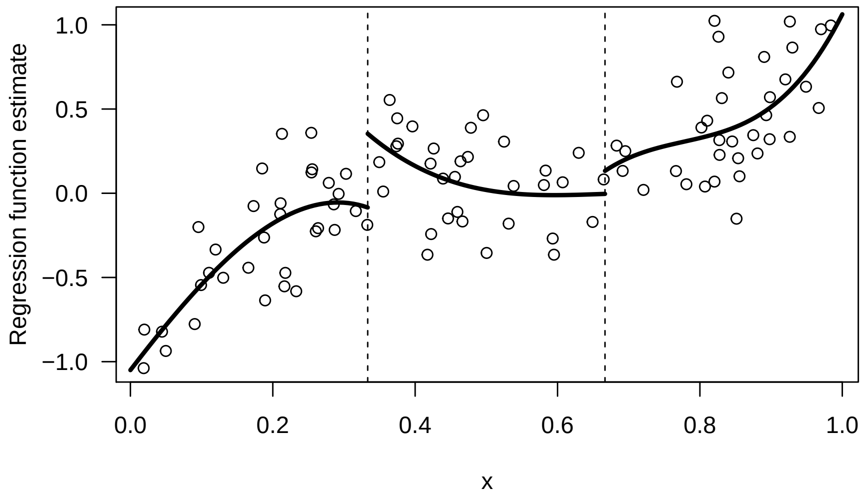


Figure 12.3: Example of a regression function estimate that is piecewise cubic within 3 bins.

- In other words, q knots defines B_{q+1} "bins" of the form $B_k = [u_{k-1}, u_{k+1})$, for $k = 2, \ldots, q$.

- Let's return to the piecewise linear estimate shown in Figure 12.2. This has two knots $u_1 = 1/3$ and $u_2 = 2/3$ and hence 3 bins.
- Also, this piecewise linear model has 6 parameters. If we let (β_{0k}, β_{1k}) denote the intercept and slope parameters for the k^{th} bin, there are 6 parameters in total because we have 3 bins, and we could write a piecewise linear model as

$$m(x) = \begin{cases} \beta_{01} + \beta_{11}x & \text{if } x < u_1 \\ \beta_{02} + \beta_{12}x & \text{if } u_1 \leq x < u_2 \\ \beta_{03} + \beta_{13}x & \text{if } x \geq u_2 \end{cases}$$

- We can make the estimated regression function look better by ensuring that it is continuous and does not have discontinuities at the knots.

- To make the estimated regression curve continuous, we just need to make sure it is continuous at the knots.
- That is, the regression coefficients need to satisfy the following two constraints:

$$\beta_{01} + \beta_{11}u_1 = \beta_{02} + \beta_{12}u_1 \qquad \text{and} \qquad \beta_{02} + \beta_{12}u_2 = \beta_{03} + \beta_{03}u_2$$

- Because we have two linear constraints, we should expect that the number of "free parameters" in a piecewise linear model with continuity constraints should equal $6 - 2 = 4$.

- Indeed, if we use the fact that under the continuity constraints: $\beta_{02} = \beta_{01} + \beta_{11}u_1 - \beta_{12}u_1$ and $\beta_{03} = \beta_{02} + \beta_{12}u_2 - \beta_{13}u_2$, then we can rewrite the piecewise linear model as

$$m(x) = \begin{cases} \beta_{01} + \beta_{11}x & \text{if } x < u_1 \\ \beta_{01} + \beta_{11}x + (\beta_{12} - \beta_{11})(x - u_1) & \text{if } u_1 \leq x < u_2 \\ \beta_{01} + \beta_{11}x + (\beta_{12} - \beta_{11})(x - u_1) + (\beta_{13} - \beta_{12})(x - u_2) & \text{if } x \geq u_2 \end{cases}$$

- We can rewrite the above more compactly as:

$$m(x) = \beta_{01} + \beta_{11}x + (\beta_{12} - \beta_{11})(x - u_1)_+ + (\beta_{13} - \beta_{12})(x - u_2)_+$$

where $(x - u_1)_+ = \max\{x - u_1, 0\}$.
- So, the functions $\varphi_0(x) = 1$, $\varphi_1(x) = x$, $\varphi_2(x) = (x - u_1)_+$, $\varphi_3(x) = (x - u_2)_+$ form a **basis** for the set of piecewise linear function with continuity constraints and knots u_1 and u_2.

- In general, a **linear spline** with q knots $u_1 < u_2 < ... < u_q$ is a funcion $m(x)$ that can be expressed as

$$m(x) = \beta_0 + \beta_1 x + \sum_{k=1}^{q} \beta_{k+1}(x - u_k)_+$$

- So, the following $q + 2$ functions form a basis for the set of linear splines with knots $u_1 < u_2 < ... < u_q$

$$\begin{aligned} \varphi_0(x) &= 1 \\ \varphi_1(x) &= x \\ \varphi_2(x) &= (x - u_1)_+ \\ \varphi_3(x) &= (x - u_2)_+ \\ &\vdots \\ \varphi_{q+1}(x) &= (x - u_q)_+ \end{aligned}$$

- Hence, if we want to fit a linear spline with q knots, we will need to estimate $q + 2$ parameters.

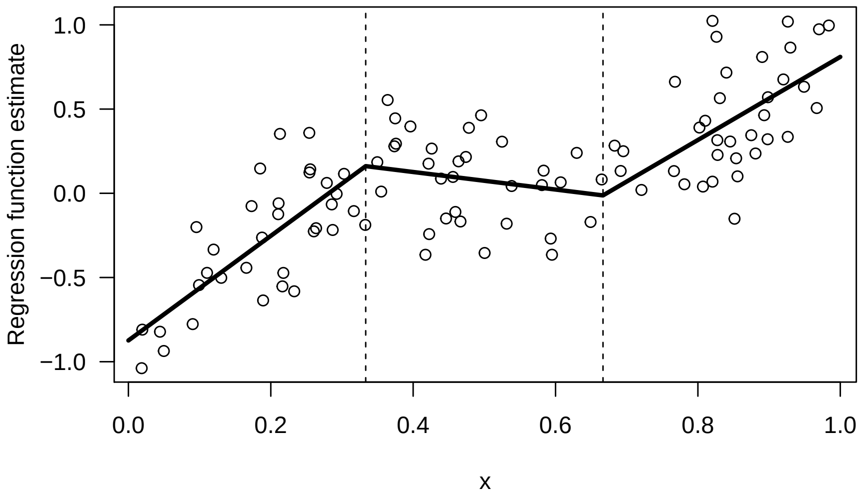


Figure 12.4: A linear spline with knots at 1/3 and 2/3. A linear spline is a piecewise linear function that is constrained to be continuous.

12.3 Cubic Splines and Regression with Splines

12.3.1 Example: Smooth Piecewise Cubic Model with 2 Knots

- Let us go back to the piecewise cubic model shown in Figure 12.3.
- This model assumes that the regression function is of the form:

$$m(x) = \begin{cases} \beta_{01} + \beta_{11}x + \beta_{21}x^2 + \beta_{31}x^3 & \text{if } 0 < x < u_1 \\ \beta_{02} + \beta_{12}x + \beta_{22}x^2 + \beta_{32}x^3 & \text{if } u_1 \le x < u_2 \\ \beta_{03} + \beta_{13}x + \beta_{23}x^2 + \beta_{33}x^3 & \text{if } 1 > x \ge u_2 \end{cases}$$

  Notice that this model has 12 parameters.

- Like in the linear spline example, if we wanted to make this piecewise cubic model continuous at the knots u_1 and u_2 we would need to impose the following constraints on the coefficients β_{jk}:

$$\begin{aligned} \beta_{01} + \beta_{11}u_1 + \beta_{21}u_1^2 + \beta_{31}u_1^3 &= \beta_{02} + \beta_{12}u_1 + \beta_{22}u_1^2 + \beta_{32}u_1^3 \\ \beta_{02} + \beta_{12}u_2 + \beta_{22}u_2^2 + \beta_{21}u_2^3 &= \beta_{03} + \beta_{13}u_2 + \beta_{23}u_2^2 + \beta_{33}u_2^3 \end{aligned} \tag{12.3}$$

- However, only forcing the piecewise cubic model to be continuous is not enough if we want a smooth estimate for the regression function that will not have obvious changes at the knots.

- We actually need the first and the second derivatives of the function to be continuous if we want a function that is smooth and does not have changes at the knots that we can detect visually.

- So, for the example that we have in Figure 12.3 with the knots u_1 and u_2, we need to enforce the additional four constraints:

$$\begin{aligned} \beta_{11} + 2\beta_{21}u_1 + 3\beta_{31}u_1^2 &= \beta_{12} + 2\beta_{22}u_1 + 3\beta_{32}u_1^2 \\ \beta_{12} + 2\beta_{22}u_2 + 3\beta_{21}u_2^2 &= \beta_{13} + 2\beta_{23}u_2 + 3\beta_{33}u_2^2 \\ 2\beta_{21} + 6\beta_{31}u_1 &= 2\beta_{22} + 6\beta_{32}u_1 \\ 2\beta_{22} + 6\beta_{21}u_2 &= 2\beta_{23} + 6\beta_{33}u_2 \end{aligned}$$

- Hence, the piecewise cubic model that has the two continuity constraints (12.3) and the four first and second derivative constraints will have $12 - 2 - 4 = 6$ parameters in total.

- In a similar way to how we derived the basis for the linear spline with knots u_1 and u_2, you can show that the following 6 functions form a basis for the set of piecewise cubic functions with knots u_1 and u_2 that are continuous and have continuous first and second derivatives:

$$\begin{aligned} \varphi_0(x) &= 1 \qquad \varphi_1(x) = x \qquad \varphi_2(x) = x^2 \qquad \varphi_3(x) = x^3 \\ \varphi_4(x) &= (x - u_1)_+^3 \qquad \varphi_5(x) = (x - u_2)_+^3 \end{aligned}$$

12.3.2 Cubic Splines

- The above example with knots u_1 and u_2 is an example of a **cubic spline**.

- **Definition**: A **cubic spline** with knots $u_1 < u_2 < ... < u_q$ is a function $f(x)$ such that
  - $f(x)$ is a cubic function over each of the intervals $(-\infty, u_1], [u_1, u_2], \dots, [u_{q-1}, u_q], [u_q, \infty)$.

  - $f(x)$, $f'(x)$, and $f''(x)$ are all continuous functions.

- One basis for the set of cubic splines with knots $u_1 < u_2 < ... < u_q$ is the following **truncated power** basis which consists of $q+4$ basis functions:

$$\begin{aligned} \varphi_0(x) &= 1 \quad \varphi_1(x) = x \quad \varphi_2(x) = x^2 \quad \varphi_3(x) = x^3 \\ \varphi_{k+3}(x) &= (x - u_k)_+^3, \quad k = 1, \ldots, q \end{aligned} \tag{12.4}$$

- A common basis for the set of cubic splines with knots $u_1 < u_2 < ... < u_q$ is the **B-spline** basis.
- Using the B-spline basis functions is mathematically equivalent to using the truncated power basis functions. Mathematically, using either basis would give you the same fitted curve.
- However, there are computational advantages to using the B-spline basis functions, and using the B-spline seems to be much more common in software implementations of spline fitting procedures.
- For a cubic spline with knots $u_1, \ldots, u_q$, we will let the following $q+4$ denote the corresponding set of B-spline basis functions

$$\varphi_{1,B}(x), \ldots, \varphi_{q+4,B}(x)$$

12.3.3 Estimating the Coefficients of a Cubic Spline

- When fitting a cubic spline, we assume that the knots $\mathbf{u} = (u_1, u_2, \ldots, u_q)$ are fixed first.
- Because the B-spline functions form a basis for the set of cubic splines with these knots, we can assume that our estimated regression function will have the form

$$m(x) = \sum_{k=1}^{q+4} \beta_k \varphi_{k,B}(x)$$

- To find the best coeffients β_k in this cubic spline model, we will minimize the residual sum-of-residuals-squared criterion

$$\sum_{i=1}^{n} \Big(Y_i - \hat{m}(x_i) \Big)^2 = \sum_{i=1}^{n} \Big(Y_i - \sum_{k=1}^{q+4} \beta_k \varphi_{k,B}(x_i) \Big)^2 \tag{12.5}$$

- Finding the coefficients $\hat{\beta}_1, \dots, \hat{\beta}_{q+4}$ that minimize this sum-of-residuals-squared criterion can be viewed as solving a regression problem with response vector $\mathbf{Y} = (Y_1, \dots, Y_n)$ and "design matrix" $\mathbf{X_u}$

$$\mathbf{X_u} = \begin{bmatrix} \varphi_{1,B}(x_1) & \varphi_{2,B}(x_1) & \dots & \varphi_{q+4,B}(x_1) \\ \varphi_{1,B}(x_2) & \varphi_{2,B}(x_2) & \dots & \varphi_{q+4,B}(x_2) \\ \vdots & \vdots & \ddots & \vdots \\ \varphi_{1,B}(x_n) & \varphi_{2,B}(x_n) & \dots & \varphi_{q+4,B}(x_n) \end{bmatrix}$$

- When written in this form, the vector of estimated regression coeffients can be expressed as

$$\begin{bmatrix} \hat{\beta}_1 \\ \hat{\beta}_2 \\ \vdots \\ \hat{\beta}_{q+4} \end{bmatrix} = (\mathbf{X_u}^T \mathbf{X_u})^{-1} \mathbf{X_u}^T \mathbf{Y}$$

and hence the vector of fitted values $\hat{\mathbf{m}} = (\hat{m}(x_1), \dots, \hat{m}(x_n))$ can be written as

$$\hat{\mathbf{m}} = \mathbf{X_u}(\mathbf{X_u}^T \mathbf{X_u})^{-1} \mathbf{X_u}^T \mathbf{Y}$$

12.3.4 An example in R

- Regression splines can be fitted in R by using the `splines` package

```
library(splines)
```

- The `bs` function from the `splines` package is useful for fitting a linear or cubic spline. This function generates the B-spline "design" matrix $\mathbf{X_u}$ described above.

```
bs(x, df, knots, degree)
```

- `x` - vector of covariates values. This can also just be the name of a variable when `bs` is used inside the `lm` function.
- `df` - the "degrees of freedom". For a cubic spline this is actually $q + 3$ rather than $q + 4$. If you just enter `df`, the `bs` function will pick the knots for you.
- `knots` - the vector of knots. If you don't want to pick the knots, you can just enter a number for the `df`.
- `degree` - the degree of the piecewise polynomial. degree=1 for a linear spline and degree=3 for a cubic spline.

- As an example of what the `bs` function returns, suppose we input the vector of covariates $(x_1, \dots, x_{10}) = (1, 2, \dots, 10)$ with knots $u_1 = 3.5$ and $u_2 = 6.5$.
- The `bs` function will return the "design matrix" $\mathbf{X_u}$ for this setup without the intercept column. When degree is 3, the dimensions of $\mathbf{X_u}$ should be 10×6 (because in this case $q = 2$). So, the `bs` function will return a 10×5 matrix (because the first column of $\mathbf{X}_u$ is dropped by `bs`):

```
xx <- 1:10
Xu <- bs(xx, knots=c(3.5, 6.5))
dim(Xu)
```

```
## [1] 10  5
```

```
head(Xu)
```

```
##              1     2       3        4 5
## [1,] 0.00000 0.000 0.00000 0.000000 0
## [2,] 0.60813 0.168 0.00808 0.000000 0
## [3,] 0.45779 0.470 0.06465 0.000000 0
## [4,] 0.17218 0.612 0.21463 0.000986 0
## [5,] 0.03719 0.515 0.42131 0.026627 0
## [6,] 0.00138 0.309 0.56631 0.123274 0
```

- Let's try an example with a linear spline to see how to use the `bs` function to fit a spline within the `lm` function.
- We will use the `bone` data again with age as the covariate. We will use the knots $\mathbf{u} = (12, 15, 18, 21, 24)$.

```
bonedat <- read.csv("~/Documents/STAT685Notes/Data/bone.csv")
knot.seq <- c(12, 15, 18, 21, 24)
linspline.bone <- lm(spnbmd ~ bs(age, knots=knot.seq, degree=1), data=bonedat)
```

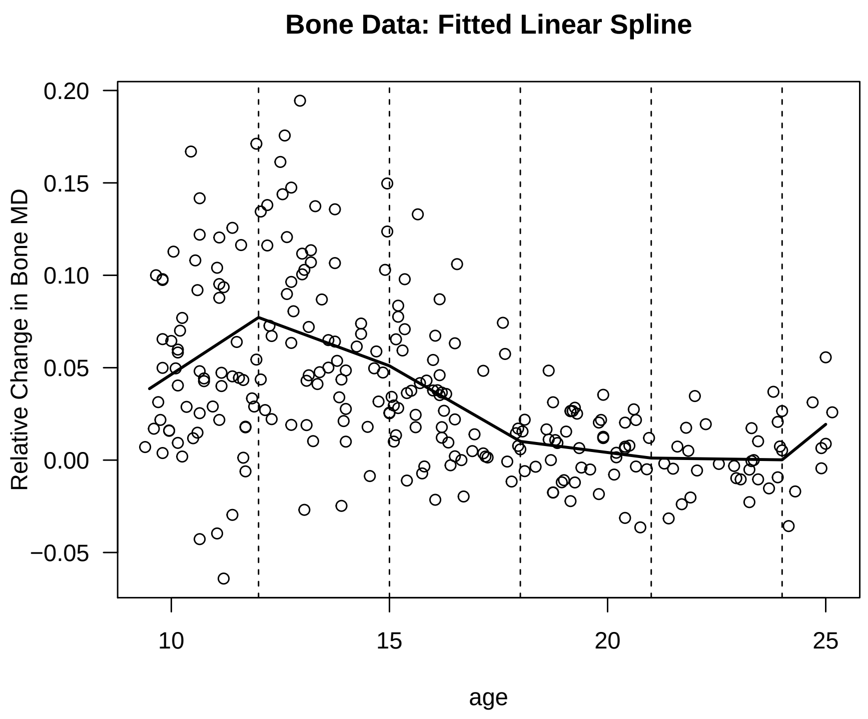
- Because we are using 5 knots, based on our discussion in Section 12.2 we should expect that there should be 7 columns in the design matrix $\mathbf{X}_u$ for this linear spline model.
- You can check this by using the following `R` code:

```
XX <- model.matrix(linspline.bone)
dim(XX)
```

```
## [1] 261   7
```

- If you want to compute the estimated spline function $\hat{m}(t_j)$ at a sequence of points $t_1, \ldots, t_l$, you can use the `predict` function on the fitted `lm` object. This is done with the following code for the points $9.5, 10, 10.5, 11, \ldots, 24.5, 25$.

```
tt <- seq(9.5, 25, by=0.5)
plot(bonedat$age, bonedat$spnbmd, xlab="age", ylab="Relative Change in Bone MD",
     main="Bone Data: Fitted Linear Spline", las=1)
lines(tt, predict(linspline.bone, data.frame(age=tt)), lwd=2)
for(k in 1:5) {
    ## Plot vertical lines at the knots
    abline(v=knot.seq[k], lty=2)
}
```



- Now let's try fitting a cubic spline model to the bone data.
- The procedure is almost exactly the same as fitting the linear spline model. We only need to change `degree=1` to `degree=3` in the `bs` function.
- We will use the same knots as we did for the linear spline.

```
knot.seq <- c(12, 15, 18, 21, 24)
cubspline.bone <- lm(spnbmd ~ bs(age, knots=knot.seq, degree=3), data=bonedat)
```

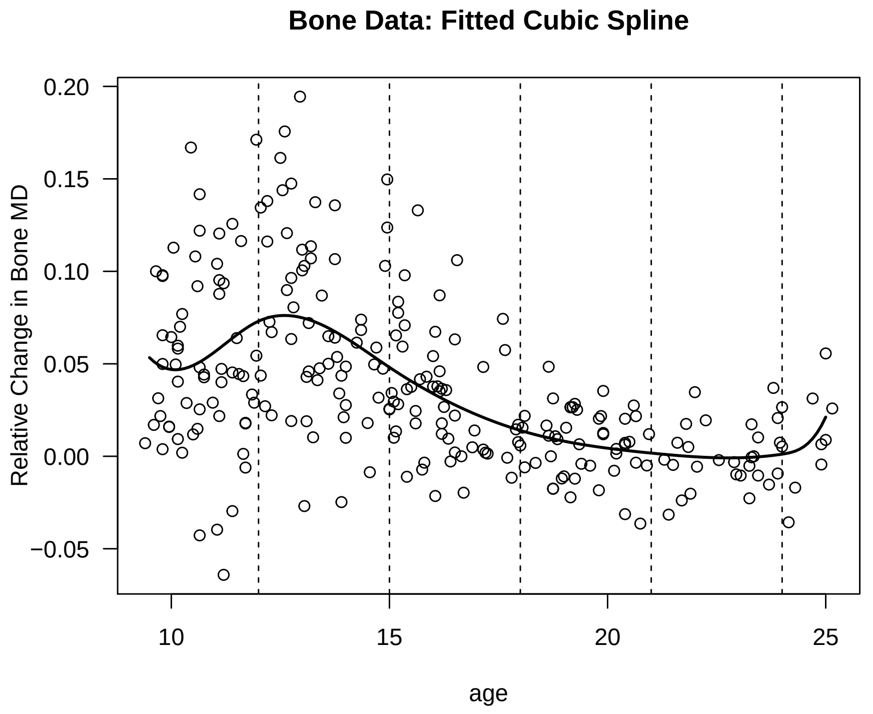
- Because we are using 5 knots, we should expect that there will be $5+4 = 9$ columns in the design matrix $\mathbf{X}_u$ for this cubic spline model.
- You can check this by using the following `R` code:

```
XX <- model.matrix(cubspline.bone)
dim(XX)
```

```
## [1] 261   9
```

- Using `predict` again, we will compute the estimated regression function $\hat{m}(x)$ at the points: $9.5, 9.6, \ldots, 24.9, 25$ and plot the result.

```
tt <- seq(9.5, 25, by=0.1)
plot(bonedat$age, bonedat$spnbmd, xlab="age", ylab="Relative Change in Bone MD",
     main="Bone Data: Fitted Cubic Spline", las=1)
lines(tt, predict(cubspline.bone, data.frame(age=tt)), lwd=2)
for(k in 1:5) {
    ## Plot vertical lines at the knots
    abline(v=knot.seq[k], lty=2)
}
```



12.3.5 Natural Cubic Splines

- Cubic splines can often have highly variable behavior near the edges of the data (i.e., for points near the smallest and largest x_i).
- One approach for addressing this problem is to use a spline which is linear for both $x < u_1$ and $x > u_q$.
- A **natural cubic spline** is a function that is still a piecewise cubic function over each of the intervals $[u_j, u_{j+1}]$ for $j = 1, \dots, q-1$, but is linear for $x < u_1$ and $x > u_q$. A natural cubic spline is still assumed to satisfy all of the continuity and derivative continuity conditions of the usual cubic spline.
- Natural cubic splines are also useful for fitting smoothing splines.

- Because we are using linear rather than cubic functions in the regions $(-\infty, u_1)$ and (u_q, ∞) this should reduce the number of necessary basis functions by 4 (so we would only need q rather $q + 4$ basis functions).

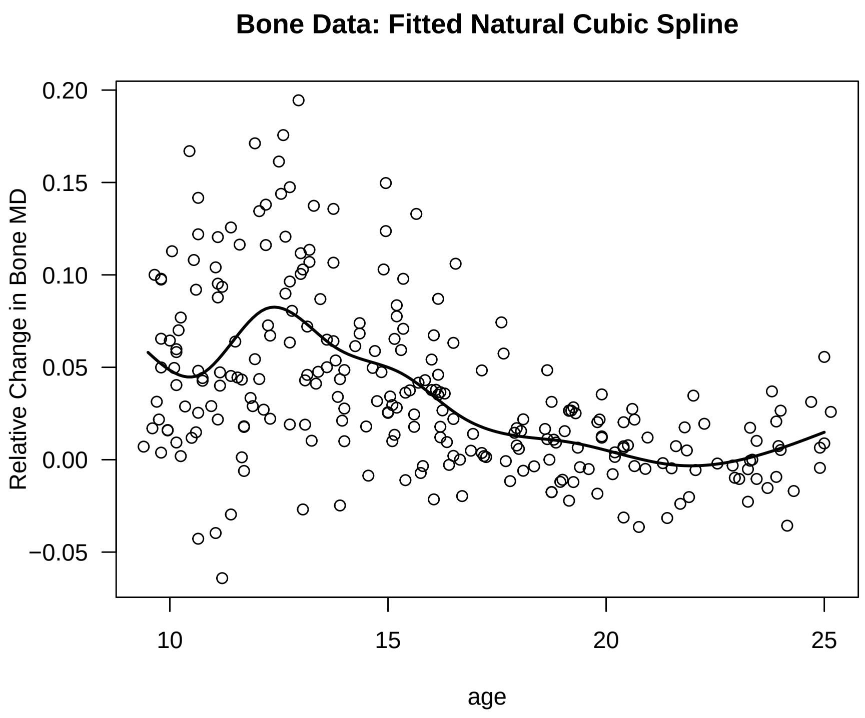
- Indeed, one basis for the set of natural cubic splines with knots $u_1, \dots, u_q$ is the following collection of q basis functions:

$$N_1(x) = 1 \qquad N_2(x) = x \qquad N_{k+2} = d_k(x) - d_{q-1}(x), k = 1, \dots, q-2,$$

where the functions $d_k(x)$ are defined as

$$d_k(x) = \frac{(x - u_k)_+^3 - (x - u_q)_+^3}{u_q - u_k}$$

```
natural.cubspline.bone <- lm(spnbmd ~ ns(age, df=8), data=bonedat)
tt <- seq(9.5, 25, by=0.1)
plot(bonedat$age, bonedat$spnbmd, xlab="age", ylab="Relative Change in Bone MD",
     main="Bone Data: Fitted Natural Cubic Spline", las=1)
lines(tt, predict(natural.cubspline.bone, data.frame(age=tt)), lwd=2)
```



12.4 Smoothing Splines

- When we used cubic splines for regression, we placed knots at a number of fixed points in between the smallest and largest x_i values.

- With what are referred to as “smoothing splines” every point x_i is treated as a potential knot, and the overall smoothness of the regression function estimate is determined through penalizing the roughness of the function.

- Motivation for smoothing splines can come from considering the following minimization problem:
  - For $\lambda \geq 0$, suppose we want to find the function $m(x)$ which solves the optimization problem:

$$\text{minimize:} \quad \sum_{i=1}^{n}\{Y_i - m(x_i)\}^2 + \lambda \int \{m''(x)\}^2 dx \tag{12.6}$$

  subject to the constraint that $m'(x)$ and $m''(x)$ are both continuous.

- The second derivative $m''(x)$ represents the curvature of m at x. So, the penalty $\int\{m''(x)\}^2 dx$ is like an average squared curvature of the function $m(x)$.
- If $\lambda = 0$, then the function $m(x)$ which minimizes (12.6) is any function such that $m(x_i) = Y_i$ for each i. This will usually be an extremely “wiggly” function.
- If $\lambda = \infty$, then we want a function $m(x)$ such that $m''(x) = 0$ for all x. The best function in this case would be a linear function $m(x) = \beta_0 + \beta_1 x$.
- By choosing $0 < \lambda < 0$, we will get a function that can be nonlinear and capture some curvature but cannot be extremely “wiggly”.

- It is possible to show that the function $\hat{m}_\lambda(x)$ which minimizes (12.6) is a **natural cubic spline** with n knots at the covariate values $(x_1, \dots, x_n)$.
- So, we can restrict our search to functions which can be written as

$$m(x) = \sum_{j=1}^{n} \beta_j N_j(x),$$

  where $N_1(x), \dots, N_n(x)$ is the set of basis functions for the set of natural cubic splines with these knots.

- So, assuming that $m(x) = \sum_{j=1}^{n} \beta_j N_j(x)$, we can re-write the minimization problem as

$$\text{minimize:}_{\beta_1,\ldots,\beta_n} \sum_{i=1}^{n} \{Y_i - \sum_{j=1}^{n} \beta_j N_j(x_i)\}^2 + \lambda \sum_{j=1}^{n}\sum_{k=1}^{n} \beta_j\beta_k \int N_j''(x) N_k''(x) dx$$

- In matrix-vector notation, this minimization problem can be written as

$$\text{minimize:}_{\beta} \quad (\mathbf{Y} - \mathbf{N}\beta)^T(\mathbf{Y} - \mathbf{N}\beta) + \lambda\beta^T\Omega\beta, \tag{12.7}$$

where $\mathbf{N}$ is the $n \times n$ matrix whose (j,k) element is $N_k(x_j)$ and Ω is the $n \times n$ matrix whose (j,k) element is $\Omega_{jk} = \int N_j''(x)N_k''(x)dx$.

- The vector of coefficients which solves the minimization problem (12.7) is given by

$$\hat{\beta} = \begin{bmatrix} \hat{\beta}_1 \\ \vdots \\ \hat{\beta}_n \end{bmatrix} = (\mathbf{N}^T\mathbf{N} + \lambda\Omega)^{-1}\mathbf{N}^T\mathbf{Y}$$

12.5 Knot/Penalty Term Selection for Splines

- For both regression splines and smoothing splines, we can write the vector of fitted values $\hat{\mathbf{m}} = (\hat{m}(x_1), \ldots, \hat{m}(x_n))$ as

$$\hat{\mathbf{m}} = \mathbf{A}\mathbf{Y},$$

for an appropriately chosen $n \times n$ matrix $\mathbf{A}$.

- For the case of a cubic regression spline with fixed knot sequence $\mathbf{u} = (u_1, \ldots, u_q)$, we have that $\hat{\mathbf{m}} = \mathbf{A}_{\mathbf{u}}\mathbf{Y}$ where

$$\mathbf{A}_{\mathbf{u}} = \mathbf{X}_{\mathbf{u}}(\mathbf{X}_{\mathbf{u}}^T\mathbf{X}_{\mathbf{u}})^{-1}\mathbf{X}_{\mathbf{u}}^T \tag{12.8}$$

Note that, in this case,

$$\text{tr}(\mathbf{A}_{\mathbf{u}}) = \text{tr}(\mathbf{X}_{\mathbf{u}}(\mathbf{X}_{\mathbf{u}}^T\mathbf{X}_{\mathbf{u}})^{-1}\mathbf{X}_{\mathbf{u}}^T) = \text{tr}((\mathbf{X}_{\mathbf{u}}^T\mathbf{X}_{\mathbf{u}})^{-1}\mathbf{X}_{\mathbf{u}}^T\mathbf{X}_{u}) = q + 4 \tag{12.9}$$

- For the case of a smoothing spline with penalty term $\lambda > 0$, we have that $\hat{\mathbf{m}} = \mathbf{A}_{\lambda}\mathbf{Y}$ where

$$\mathbf{A}_{\lambda} = \mathbf{N}(\mathbf{N}^T\mathbf{N} + \lambda\Omega)^{-1}\mathbf{N}^T$$

12.5.1 The Cp Statistic

- As with kernel and local regression method described in Chapter 11, the C_p statistic is defined as the mean residual sum of squares plus a penalty which depends on the matrix $\mathbf{A}$.

- In the context of regression splines where $\hat{\mathbf{m}} = \mathbf{A_u Y}$, the C_p statistic can be written as:

$$\begin{aligned} C_p(q) &= \frac{1}{n}\sum_{i=1}^{n}\{Y_i - \hat{m}(x_i)\}^2 + \frac{2\hat{\sigma}^2}{n}\text{tr}(\mathbf{A}_u) \\ &= \frac{1}{n}\sum_{i=1}^{n}\{Y_i - \hat{m}(x_i)\}^2 + \frac{2\hat{\sigma}^2(q+4)}{n} \end{aligned}$$

- In the context of smoothing splines where $\hat{\mathbf{m}} = \mathbf{A}_\lambda \mathbf{Y}$, the C_p statistic can be written as

$$\begin{aligned} C_p(\lambda) &= \frac{1}{n}\sum_{i=1}^{n}\{Y_i - \hat{m}(x_i)\}^2 + \frac{2\hat{\sigma}^2}{n}\text{tr}(\mathbf{A}_\lambda) \\ &= \frac{1}{n}\sum_{i=1}^{n}\{Y_i - \hat{m}(x_i)\}^2 + \frac{2\hat{\sigma}^2}{n}\text{tr}\Big((\mathbf{N}^T\mathbf{N} + \lambda\Omega)^{-1}\mathbf{N}^T\mathbf{N}\Big) \end{aligned}$$

12.5.2 Leave-one-out Cross-Validation

- As mentioned in Chapter 11, the leave-one-out cross-validation can be expressed as a weighted sum-of-squared residuals with the weights coming from the diagonals of the "smoothing" matrix $\mathbf{A}$.

- For the smoothing spline, the leave-one-out cross-validation criterion is

$$\text{LOOCV}(\lambda) = \frac{1}{n}\sum_{i=1}^{n}\Big(\frac{Y_i - \hat{m}_\lambda(x_i)}{1 - a_i^\lambda(x_i)}\Big)^2$$

where $a_i^\lambda(x_i)$ denotes the i^{th} diagonal of the matrix $\mathbf{A}_\lambda$.

12.5.3 Generalized Cross-Validation

- A criterion which we did not mention in Chapter 11 is Generalized Cross-Validation (GCV).

- For the smoothing parameter λ, the GCV criterion is defined as

$$\text{GCV}(\lambda) = \frac{1}{n}\sum_{i=1}^{n}\Big(\frac{Y_i - \hat{m}_\lambda(x_i)}{1 - \text{tr}(\mathbf{A}_\lambda)/n}\Big)^2 = \Big(\frac{n}{n - \text{tr}(\mathbf{A}_\lambda)}\Big)^2 \frac{1}{n}\sum_{i=1}^{n}\{Y_i - \hat{m}_\lambda(x_i)\}^2$$

- The GCV criterion can be seen as replacing the individual diagonal elements in the LOOCV criterion with their average value $\text{tr}(\mathbf{A}_\lambda)/n$.

- GCV can often perform better when some of the diagonal elements of $\mathbf{A}_\lambda$ are close to 1.

12.6 Fitting Smoothing Splines in R

- The R function `smooth.spline` will fit smoothing splines.

```
smooth.spline(x, y, df, lambda, cv=FALSE)
```

- **x** - the vector of covariate values.
- **y** - the vector of responses.
- **df** - the trace of the "smoother" matrix. This is the trace of the matrix $\mathbf{A}_\lambda = \mathbf{N}(\mathbf{N}^T\mathbf{N} + \lambda\Omega)^{-1}\mathbf{N}^T\mathbf{Y}$. This must be less than or equal to n (where n here is the number of unique values of the x_i).
- **lambda** - the value of λ in the matrix $\mathbf{A}_\lambda = \mathbf{N}(\mathbf{N}^T\mathbf{N} + \lambda\Omega)^{-1}\mathbf{N}^T\mathbf{Y}$. Note that you should only enter one of the **df** or **lambda** arguments.
- **cv** - the smooth.spline function will perform cross-validation whenever both the **df** and **lambda** arguments are left empty. When both of these arguments are left empty and `cv=FALSE`, the function will use GCV to select the smoothing parameter. When both **df** and **lambda** are left empty and `cv = TRUE`, the function will use LOOCV to select the smoothing parameter.
- Notice that when only input the `x` and `y` vectors, the `smooth.spline` will automatically use generalized cross-validation to select the smoothing parameter.

- To see how to use the `smooth.spline` function and how to use the C_p statistic, LOOCV, or GCV for selecting the smoothing parameter in smoothing splines or the number of knots in regression splines, we will again consider the `bone` data.
- To start, let's use the `smooth.spline` function on the bone data using GCV to find the smoothing parameter:

```
ss.bone <- smooth.spline(x=bonedat$age, y=bonedat$spnbmd)

plot(bonedat$age, bonedat$spnbmd, las=1, pch=16, xlab="age",
     ylab="Relative Change in Bone MD", main="Bone Data: Smoothing Spline using GCV")
lines(ss.bone$x, ss.bone$y, lwd=3, col="red")
```

- If you just type in `ss.smooth`, this will show the degrees of freedom used, the value of λ used, and the value of the GCV criterion for the chosen λ

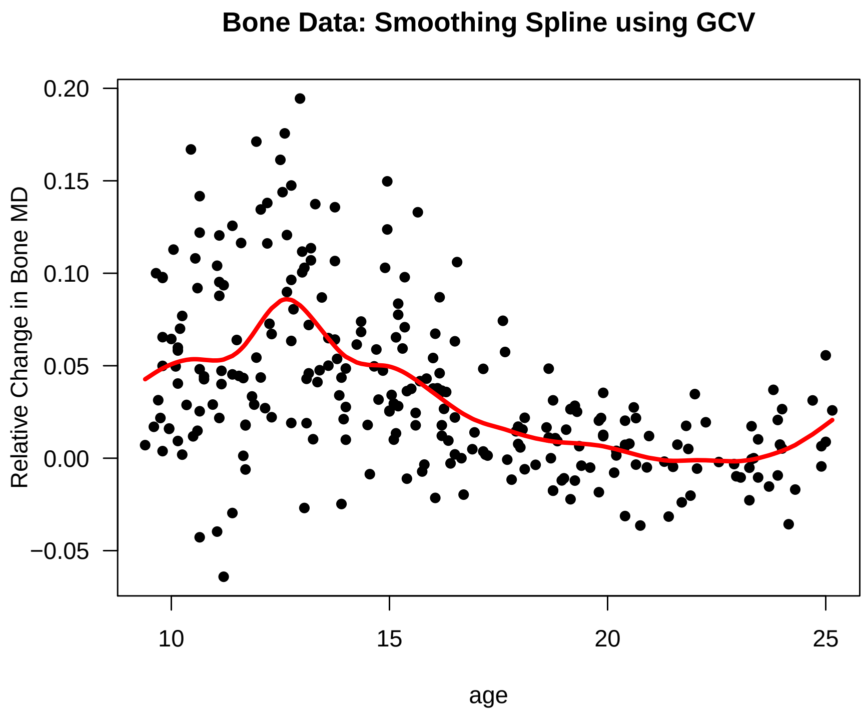


Figure 12.5: Smoothing spline fit for the bone data. This used the smooth.spline function with all the default settings.

```
ss.bone
```

```
## Call:
## smooth.spline(x = bonedat$age, y = bonedat$spnbmd)
##
## Smoothing Parameter  spar= 0.712  lambda= 0.000239 (15 iterations)
## Equivalent Degrees of Freedom (Df): 12.2
## Penalized Criterion (RSS): 0.222
## GCV: 0.00152
```

- According to the GCV criterion, the best value for λ is about $\lambda^* \approx 0.0002$ and the corresponding value for the degrees of freedom is $df^* = \text{tr}(\mathbf{A}_{\lambda^*}) \approx$ 12.15.
- Using the LOOCV criterion, the best value for the degrees of freedom is 13.

```
ss.bone2 <- smooth.spline(x=bonedat$age, y=bonedat$spnbmd, cv=TRUE)
```

```
## Warning in smooth.spline(x = bonedat$age, y = bonedat$spnbmd, cv = TRUE): cross-
## validation with non-unique 'x' values seems doubtful
```

```
ss.bone2
```

```
## Call:
## smooth.spline(x = bonedat$age, y = bonedat$spnbmd, cv = TRUE)
##
## Smoothing Parameter  spar= 0.694  lambda= 0.000177 (13 iterations)
## Equivalent Degrees of Freedom (Df): 13
## Penalized Criterion (RSS): 0.219
## PRESS(l.o.o. CV): 0.00149
```

12.6.1 Smoothing Parameter Selection for the Smoothing Spline with the Cp statistic

- Suppose we wanted to find the best value of $\text{tr}(\mathbf{A}_\lambda)$ of the smoothing spline using the C_p statistic.
- The first thing we want to find is an estimate of σ^2 that we can keep fixed across different values of the smoothing parameter. Let's use the same value of $\hat{\sigma}^2 = 0.0015$ that we used in Chapter 11:

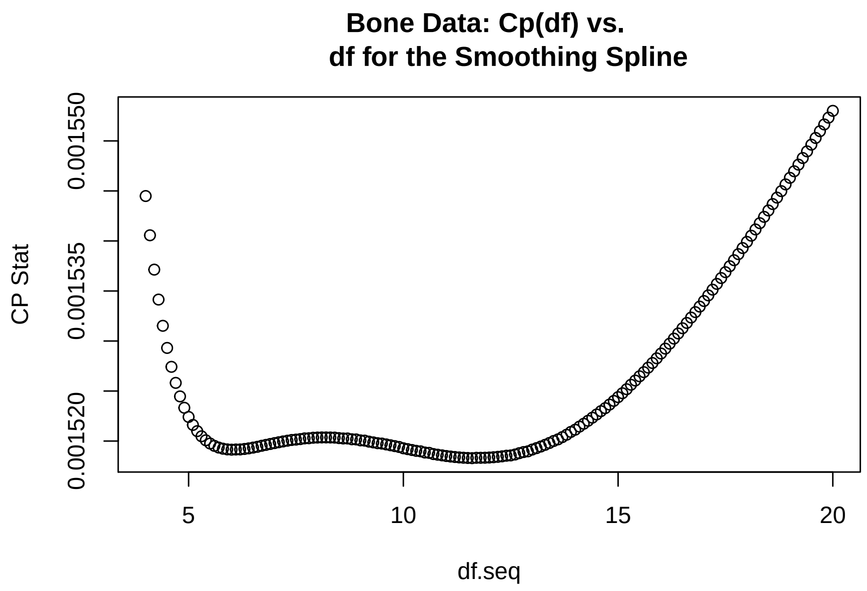
```
sigsq.est <- 0.0015
```

- Now, let's write a function that computes the C_p statistic for a smoothing spline model given that we input the data, the degrees of freedom $\mathrm{tr}(\mathbf{A}_\lambda)$, and $\hat{\sigma}^2$.

```
CpStatSmoothSpline <- function(x, y, df, sigsq.hat) {
  n <- length(x)
  ss.obj <- smooth.spline(x=x, y=y, df=df)
  ## Now, compute the vector of residuals from this fitted smoothing spline
  residu <- y - fitted(ss.obj)
  ans <- mean(residu^2) + (2*sigsq.hat/n)*df
  return(ans)
}
```

- Now, compute the C_p statistic for values of the degrees of freedom between 4 and 20. From the plot of the C_p vs. the degrees of freedom it looks like the best value for the degrees of freedom is about 12

```
df.seq <- seq(4, 20, by=.1)
nx <- length(df.seq)
Cp.seq <- numeric(nx)
for(k in 1:nx) {
    Cp.seq[k] <- CpStatSmoothSpline(x=bonedat$age, y=bonedat$spnbmd, df=df.seq[k],
                                    sigsq.hat=sigsq.est)
}

plot(df.seq, Cp.seq, ylab="CP Stat", main="Bone Data: Cp(df) vs.
     df for the Smoothing Spline")
```



- More specifically, it's about 11.6:

```
df.seq[which.min(Cp.seq)]
```

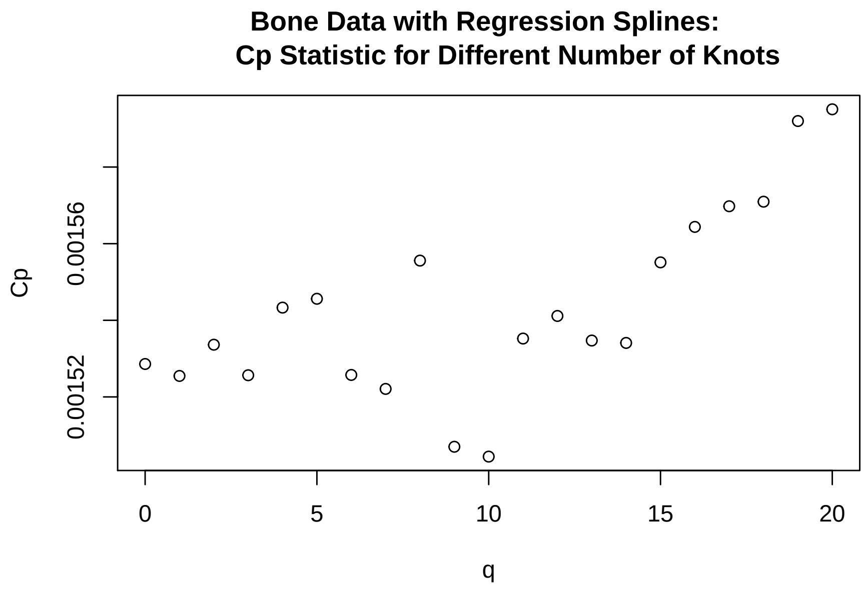
```
## [1] 11.6
```

12.6.2 Knot Selection for Regression Splines with the Cp statistic

- Let's go back to the regression spline methods we described in Section 12.3 also use the bone data to try to find the best number of knots for the case of regression splines.
- We will consider knots $u_1, \ldots, u_q$ that are computed automatically by the `bs` function when we specify the degrees of freedom. These are based on the quantiles of the covariate x_i. We will consider values of q between 0 and 20.

```
sigsq.hat <- .0015
n <- nrow(bonedat)

qqseq <- 0:20
Cp.seq <- rep(0, length(qqseq))
```

```
nq <- length(qqseq)
for(k in 1:nq) {
  q = qqseq[k]
  tt <- 1:q

  #uu <- 9.4 + (15.7/(q+1))*tt
  if(q == 0) {
    tmp <- lm(bonedat$spnbmd ~ bs(bonedat$age, df=q+3))
  } else {
    tmp <- lm(bonedat$spnbmd ~ bs(bonedat$age, df=q+3))
  }
  RSS <- mean((bonedat$spnbmd - tmp$fitted.values)^2)
  Cp.seq[k] <- RSS + (2*sigsq.hat*(q + 4))/n
}
plot(qqseq, Cp.seq, xlab="q", ylab="Cp", main="Bone Data with Regression Splines:
     Cp Statistic for Different Number of Knots")
```



- From the plot, it looks like the best number of knots when using the C_p statistic is $q = 10$.

Part V

Nonparametric Regression: Part II

Chapter 13

Regression Trees and CART

13.1 Introduction

- Let's think about the regressogram estimate again.
- The regressogram estimate of the regression function is a piecewise constant function that is constant within each of p "bins"

$$\hat{m}^{R}_{h_n}(x) = \frac{1}{a}\sum_{i=1}^{n} Y_i I(x_i \in B_k), \qquad \text{if } x \in B_k$$

- Figure 13.1 shows an example of a regressogram estimate with 3 bins.

![Regressogram estimate with the 3 bins: [0,1/3), [1/3, 2/3), 2/3, 1).

- Suppose we were forced to combined two adjacent bins to estimate the regression function. For the data shown in 13.1, which two bins should we combine if we were forced to do so? The only options here are to combine bins 1 and 2 or to combine bins 2 and 3.
- I would say we should combine bins 1 and 2. Look at Figures 13.1 and 13.1 for a comparison of these two choices.
- The responses Y_i change much more over the range of the third bin than they do over the first and second bins. Hence, an intercept model for the first two bins is not all that bad.

- In contrast, an intercept model for the last two bins is a terrible model.

![Regressogram estimate with the 2 bins: [0,1/3), 1/3, 1).

![Regressogram estimate with the 2 bins: [0,2/3), 2/3, 1).

- The intuition for why the choice of bins $[0, 2/3), [2/3, 1)$ is better than the choice of bins $[0, 1/3), [1/3, 1)$ can be formalized by considering the within-bin variation of Y_i.
- For two bins B_1 and B_2, the within-bin sum of squares (WBSS) of Y_i is

$$\text{WBSS} = \sum_{k=1}^{2}\sum_{i=1}(Y_i - \bar{Y}_k)^2 I(x_i \in B_k)$$

  where $\bar{Y}_k = \frac{1}{n_k}\sum_{i=1}^{n} Y_i$ denotes the mean of the responses within the k^{th} bin.
- You want to choose the bins in order to minimize the within-bin sum of squares. The reason for this is that: if the within-bin sum of squares is low, an intercept model for each bin will fit the data very well.

- For the data shown in Figures 13.1 - 13.1, the WBSS when using the bins $[0, 1/3), [1/3, 1)$ is

```
sum( (yy[xx < 1/3] - mean(yy[xx < 1/3]))^2 ) +
  sum( (yy[xx >= 1/3] - mean(yy[xx >= 1/3]))^2 )
```

```
## [1] 62.72984
```

- The WBSS when using the bins $[0, 2/3), [2/3, 1)$ is

```
sum( (yy[xx < 2/3] - mean(yy[xx < 2/3]))^2 ) +
  sum( (yy[xx >= 2/3] - mean(yy[xx >= 2/3]))^2 )
```

```
## [1] 20.19249
```

13.2 Regression Trees with a Single Covariate

- For a single covariate, regression trees estimate the regression function by a piecewise constant function that is constant within each of several bins. We will focus on the well-known CART (Classification and Regression Trees) method for using regression trees.
- More generally, with multivariate covariates CART will fit a regression function that is constant within each of many multi-dimensional "rectangles".
- The main difference between CART and the regressogram is that the placements and widths of the bins in CART are chosen in a more selective manner than the regressogram.
- Specifically, rather than just using a collection of bins of fixed width, CART chooses where to place the bin boundaries by considering the resulting within-bin sum of squares.

- CART constructs the bins through sequential binary splits of the x axis.
- In the first step, CART will divide the covariates into two bins B_1 and B_2. These bins will have the form $B_1 = (-\infty, t_1)$ and $B_2 = [t_1, \infty)$.
- At the next step, CART will create 4 bins by further dividing each of these two bins into two more bins. So, we will have four bins $B_{11}, B_{12}, B_{21}, B_{22}$.
- Bins B_{11} and B_{12} will have the form $B_{11} = (-\infty, t_{11})$ and $B_{12} = [t_{11}, t_1)$, and bins B_{21} and B_{22} will have the form $B_{21} = [t_1, t_{21})$ and $B_{22} = [t_{21}, \infty)$.
- You can repeat this process to get smaller and smaller bins. Usually, this process will stop once a threshold for the minimum number of observations in a bin has been reached.

- This sequential process for constructing bins is typically depicted with a binary decision tree.
- Figure 13.2 shows the decision tree representation of a CART regression function estimate with 4 bins.

![Binary decision tree representing a regression function estimate with 4 bins where it is assumed that all the covariates are between 0 and 1. The 4 bins here are [0, 0.6), [0.6, 0.74), [0.74, 0.89), 0.89, 1).

- Figure 13.1 shows the regression function estimate which corresponds to the decision tree shown in Figure 13.2.

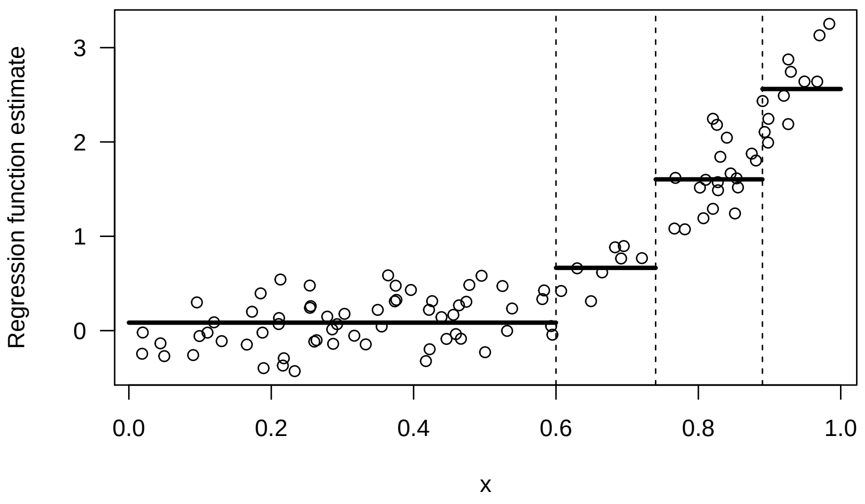


Figure 13.1: Regression function estimate that corresponds to the decision tree shown in the previous figure.

13.2.1 Determining the Split Points

- The first two bins are determined by the "split point" t_1. To find this split point, CART looks at the within-bin sum of squares induced by a splitting point t.

$$\begin{aligned} \text{WBSS}(t) &= \sum_{k=1}^{2}\sum_{i=1}^{n}(Y_i - \bar{Y}_k)^2 I(x_i \in B_k) \\ &= \sum_{i=1}^{n}(Y_i - \bar{Y}_1)^2 I(x_i < t) + \sum_{i=1}^{n}(Y_i - \bar{Y}_2)^2 I(x_i \geq t) \end{aligned}$$

- The first split point t_1 is the value of t which minimizes this within-bin sum of squares criterion. That is,

$$t_1 = \text{argmin}_t \ \text{WBSS}(t) = \text{argmin}_{t \in \{x_1, \ldots, x_n\}} \ \text{WBSS}(t)$$

- To find t_1, we only have to take the minimum over the set of covariates since the value of WBSS(t) only changes at each x_i.

- Figure 13.2 shows a plot of WBSS(t) vs. t for the data shown in Figures 13.1 - 13.1.

- Figure 13.2 suggests that the value of t_1 will be around 0.75.

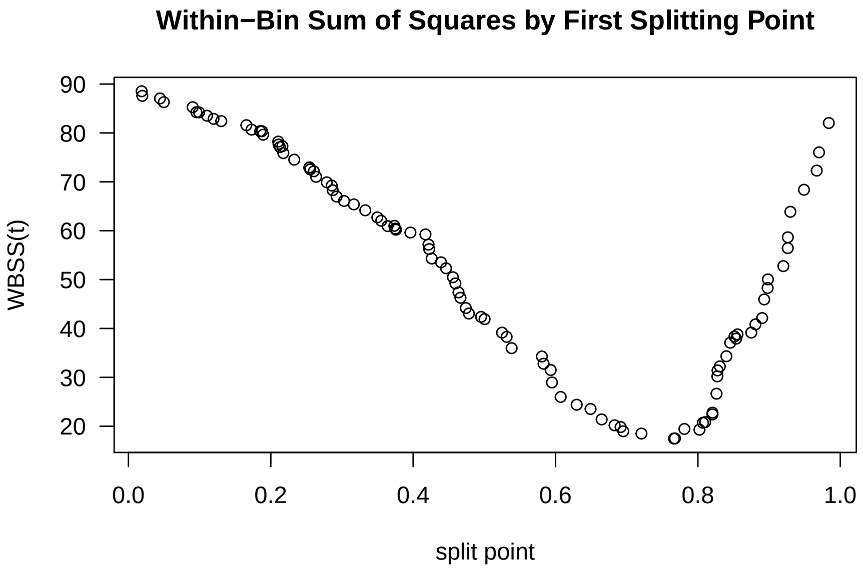


Figure 13.2: Plot of WBSS(t) vs. t for the data shown in the above figures.

- The splitting point "partitions" the data into two datasets. The indices of these datasets are defined as

$$\begin{aligned}\mathcal{D}_1 &= \big\{(Y_i, x_i) : x_i < t_1\big\} \\ \mathcal{D}_2 &= \big\{(Y_i, x_i) : x_i \geq t_1\big\}\end{aligned}$$

- After finding t_1, we can find the next two splitting points t_{11} and t_{21} by using the exact same procedure we used to find t_1.

- That is, t_{11} and t_{21} are given by

$$\begin{aligned}t_{11} &= \text{argmin}_t \text{ WBSS}_1(t) \\ t_{21} &= \text{argmin}_t \text{ WBSS}_2(t)\end{aligned}$$

  where $\text{WBSS}_a(t)$ is the within-bin sum of squares for dataset $\mathcal{D}_a$:

$$\text{WBSS}_a(t) = \sum_{i \in \mathcal{D}_a} (Y_i - \bar{Y}_{1a})^2 I(x_i < t) + \sum_{i \in \mathcal{D}_a} (Y_i - \bar{Y}_{2a})^2 I(x_i \geq t)$$

- The splitting points t_{11} and t_{21} will further partition the dataset into 4 datasets. Additional splitting points $t_{12}, t_{22}, t_{32}, t_{42}$ which further partition the dataset, can be found by minimizing the within-bin sum of squares for each of these 4 datasets.
- This algorithm for constructing smaller and smaller bins is often referred to as recursive partitioning.

13.3 Regression Trees With Multiple Covariates

- When we have multivariate covariates where $\mathbf{x}_i = (x_{i1}, \dots, x_{ip})$, CART will partition the covariate space into multivariate "rectangles".
- An example of a CART-type regression function estimate for the case of two covariates is shown in Figure 13.3

- One advantage of CART is that it can easily handle covariates of different types, for example, both continuous and binary covariates.
- You can see an example of this by looking at the `bone` data. This dataset has both sex and age as covariates.
- A CART regression tree for the bone data using both age and sex is shown in Figure 13.4.

- Figure 13.5 plots the regression function estimate which corresponds to the decision tree shown in Figure 13.4. Notice that the regression function estimate for men vs. women only differs for ages < 12.

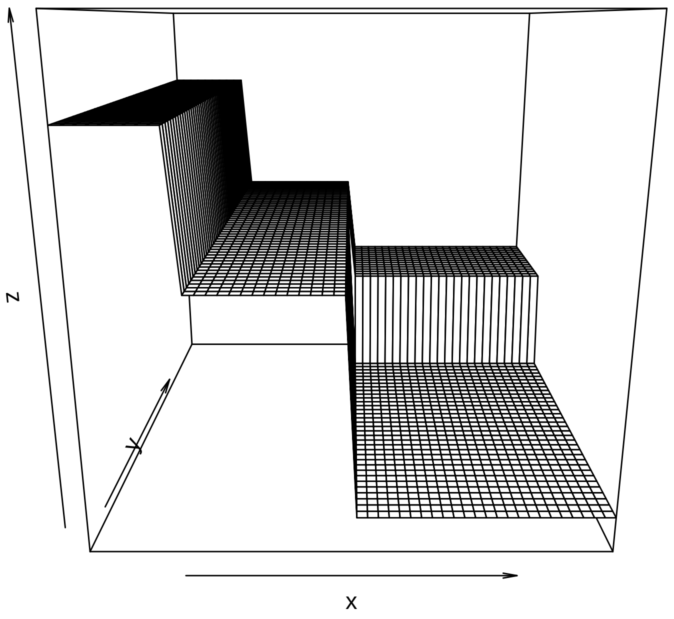


Figure 13.3: An example of a CART-type estimate of a regression function that has two covariates.

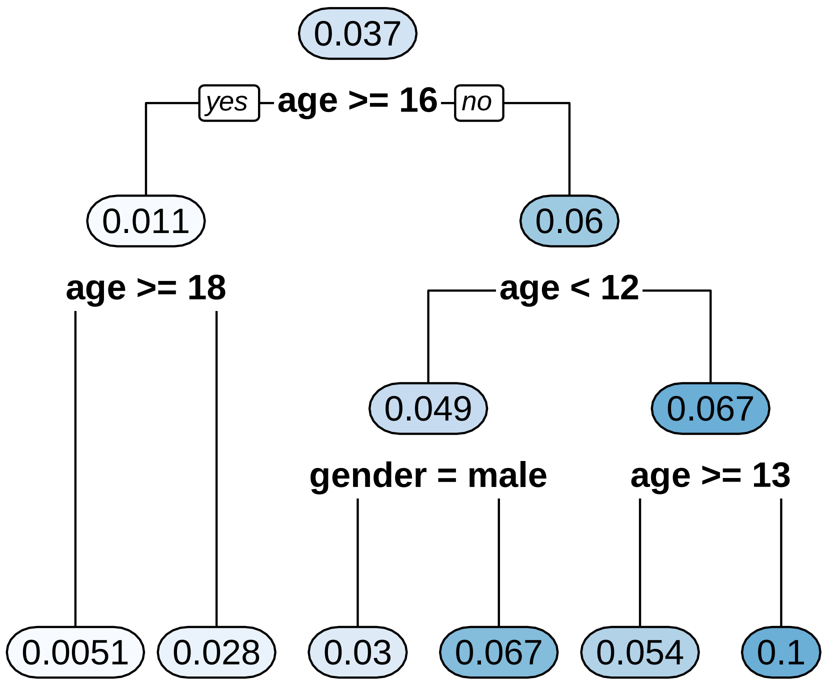


Figure 13.4: Regression tree for the bone data. This fitted regression tree has 6 bins.

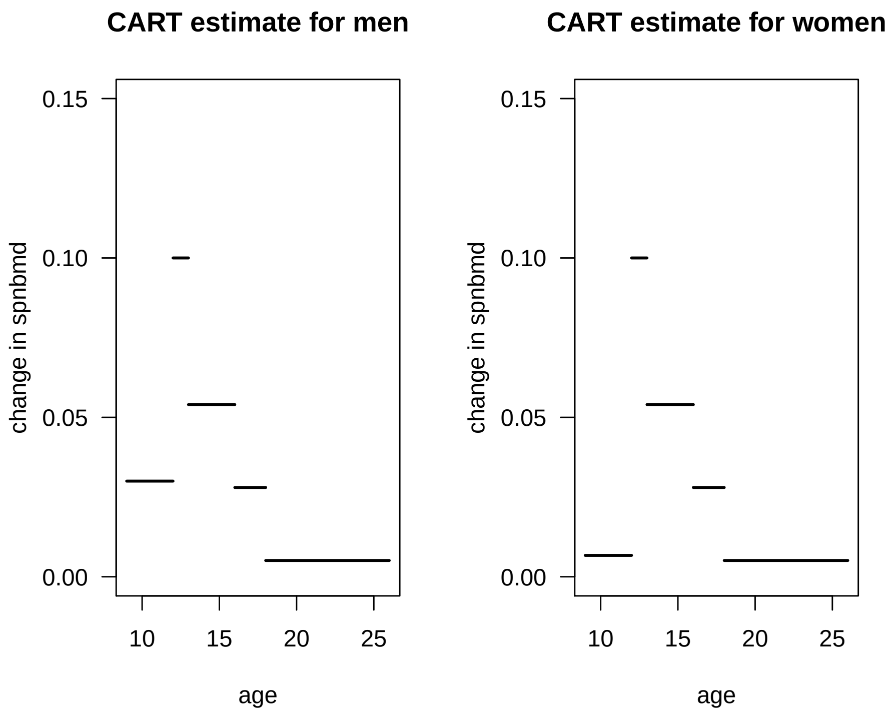


Figure 13.5: Plot of regression function estimate that corresponds to the decision tree in the previous figure.

Bibliography

Altmann, A., Toloşi, L., Sander, O., and Lengauer, T. (2010). Permutation importance: a corrected feature importance measure. *Bioinformatics*, 26(10):1340–1347.

Bühlmann, P. (2002). Bootstraps for time series. *Statistical science*, pages 52–72.

Casella, G. and Berger, R. L. (2002). *Statistical inference*, volume 2. Duxbury Pacific Grove, CA.

Chen, Y.-C. (2017). A tutorial on kernel density estimation and recent advances. *Biostatistics & Epidemiology*, 1(1):161–187.

Davison, A. C. and Hinkley, D. V. (1997). *Bootstrap methods and their application*, volume 1. Cambridge university press.

Divine, G. W., Norton, H. J., Barón, A. E., and Juarez-Colunga, E. (2018). The wilcoxon–mann–whitney procedure fails as a test of medians. *The American Statistician*, 72(3):278–286.

Friedman, J. H. (1984). A variable span scatterplot smoother. *Laboratory for Computational Statistics, Stanford University Technical Report No. 5.*

Härdle, W. K., Müller, M., Sperlich, S., and Werwatz, A. (2012). *Nonparametric and semiparametric models.* Springer Science & Business Media.

Hollander, M., Wolfe, D. A., and Chicken, E. (2013). *Nonparametric statistical methods*, volume 751. John Wiley & Sons.

Huo, X. and Székely, G. J. (2016). Fast computing for distance covariance. *Technometrics*, 58(4):435–447.

Izenman, A. J. (2008). Modern multivariate statistical techniques. *Regression, classification and manifold learning*, 10:978–0.

Lehmann, E. L. and Romano, J. P. (2006). *Testing statistical hypotheses.* Springer Science & Business Media.

Nembrini, S. (2019). On what to permute in test-based approaches for variable importance measures in random forests. *Bioinformatics*, 35(15):2701–2705.

Scott, D. W. (1979). On optimal and data-based histograms. *Biometrika*, 66(3):605–610.

Shao, J. (2003). *Mathematical Statistics*. Springer-Verlag New York Inc, 2nd edition.

Sheather, S. J. (2004). Density estimation. *Statistical science*, pages 588–597.

Silverman, B. W. (2018). *Density estimation for statistics and data analysis*. Routledge.

Székely, G. J., Rizzo, M. L., Bakirov, N. K., et al. (2007). Measuring and testing dependence by correlation of distances. *The annals of statistics*, 35(6):2769–2794.

Tsybakov, A. B. (2008). *Introduction to nonparametric estimation*. Springer Science & Business Media.

Van der Vaart, A. W. (2000). *Asymptotic statistics*, volume 3. Cambridge university press.

Wasserman, L. (2006). *All of nonparametric statistics*. Springer Science & Business Media.