CS 660: Mathematical Foundations of Analytics

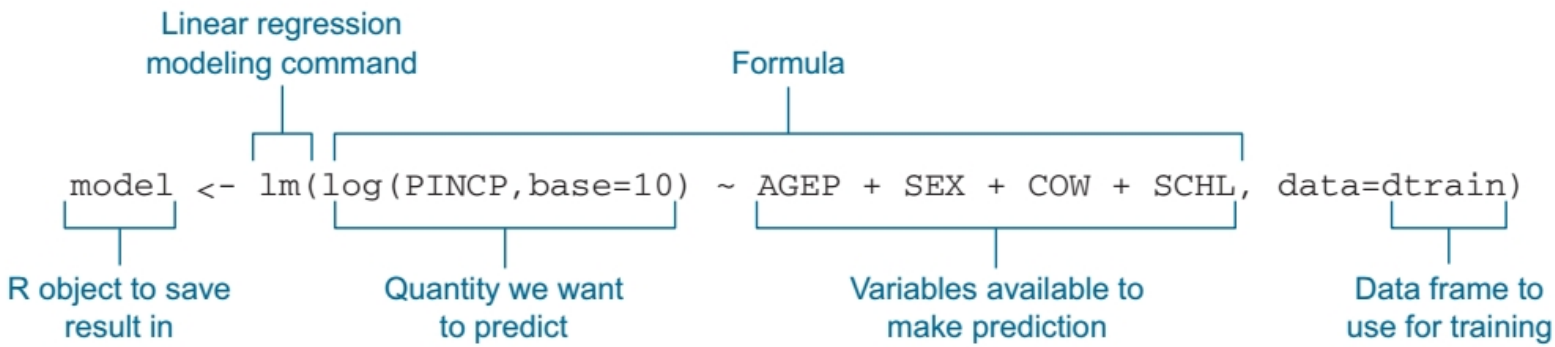
Dr. Francis Parisi

Pace University

Spring 2018

Regression Models in R

- Fitting a regression model in R



Regression Models in R

Symbol	Usage
`~`	Separates response variables on the left from the explanatory variables on the right. For example, a prediction of `y` from `x`, `z`, and `w` would be coded `y ~ x + z + w`.
`+`	Separates predictor variables.
`:`	Denotes an interaction between predictor variables. A prediction of `y` from `x`, `z`, and the interaction between `x` and `z` would be coded `y ~ x + z + x:z`.
`*`	A shortcut for denoting all possible interactions. The code `y ~ x * z * w` expands to `y ~ x + z + w + x:z + x:w + z:w + x:z:w`.
`^`	Denotes interactions up to a specified degree. The code `y ~ (x + z + w)^2` expands to `y ~ x + z + w + x:z + x:w + z:w`.
`.`	A placeholder for all other variables in the data frame except the dependent variable. For example, if a data frame contained the variables `x`, `y`, `z`, and `w`, then the code `y ~ .` would expand to `y ~ x + z + w`.
`-`	A minus sign removes a variable from the equation. For example, `y ~ (x + z + w)^2 - x:w` expands to `y ~ x + z + w + x:z + z:w`.
`-1`	Suppresses the intercept. For example, the formula `y ~ x -1` fits a regression of `y` on `x`, and forces the line through the origin at `x=0`.
`I()`	Elements within the parentheses are interpreted arithmetically. For example, `y ~ x + (z + w)^2` would expand to `y ~ x + z + w + z:w`. In contrast, the code `y ~ x + I((z + w)^2)` would expand to `y ~ x + h`, where `h` is a new variable created by squaring the sum of `z` and `w`.
`function`	Mathematical functions can be used in formulas. For example, `log(y) ~ x + z + w` would predict `log(y)` from `x`, `z`, and `w`.

Regression Models in R

Function	Action
`summary()`	Displays detailed results for the fitted model
`coefficients()`	Lists the model parameters (intercept and slopes) for the fitted model
`confint()`	Provides confidence intervals for the model parameters (95% by default)
`fitted()`	Lists the predicted values in a fitted model
`residuals()`	Lists the residual values in a fitted model
`anova()`	Generates an ANOVA table for a fitted model, or an ANOVA table comparing two or more fitted models
`vcov()`	Lists the covariance matrix for model parameters
`AIC()`	Prints Akaike's Information Criterion
`plot()`	Generates diagnostic plots for evaluating the fit of a model
`predict()`	Uses a fitted model to predict response values for a new dataset

Regression Models in R

1. Using the database `women` in the base R installation fit a regression of `weight` on `height` (don't forget to save your model)
2. Display a summary of your model, and interpret the results
3. Display the fitted values
4. Display the residuals
5. Plot `weight` (y-axis) and `height` (x-axis)

Regression Models in R

A polynomial regression is a regression model where we use use powers of the explanatory variables

Refit the the regression as a quadratic model and interpret the results as before

```
fit2 <- lm(weight ~ height + I(height^2), data=women)
```

Note this is still a linear model even though we have a quadratic term – it's a linear combination of the $\hat{\beta}'s$

An example of a nonlinear model is

$$\hat{Y} = \hat{\beta}_0 + \hat{\beta}_1 e^{X/\hat{\beta}_2}$$

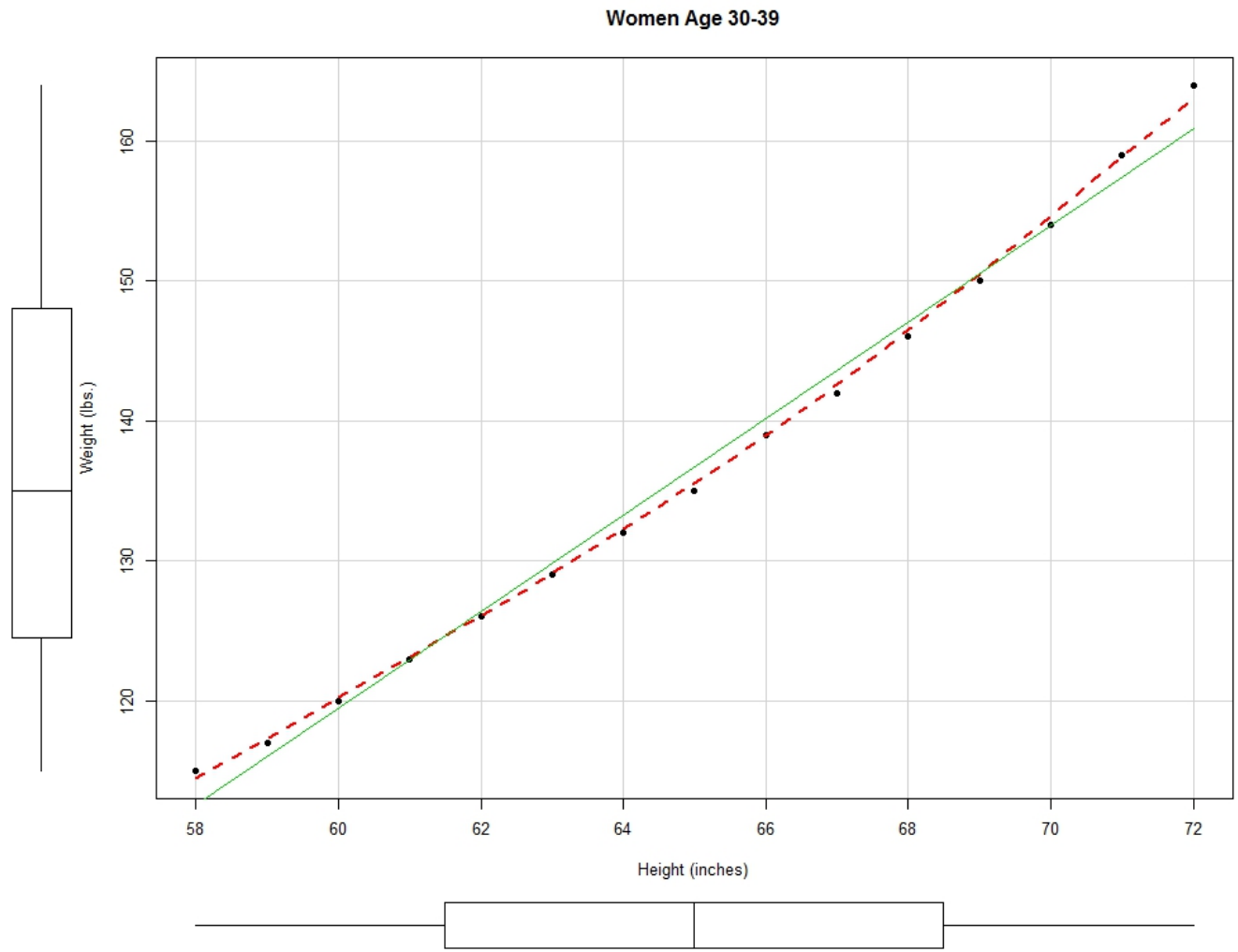
You can fit nonlinear models with `nls()`

Regression Models in R

The `car` library has a `scatterplot()` function that can create an informative plot

```
library(car)
scatterplot(weight ~ height, data=women,
  spread=FALSE, smoother.args=list(lty=2), pch=19,
  main="Women Age 30-39",
      xlab="Height (inches)",
  ylab="Weight (lbs.)")
```

Regression Models in R



Regression Models in R

- When working with several independent variables we need to check for correlation among them
- This gives insight into possible interactions and multicollinearity
- We'll us the built-in data set `state.x77`

Regression Models in R

- Since `state.x77` is a matrix we need to convert it to a data.frame

```
states <-
as.data.frame(state.x77[,c("Murder",
"Population", "Illiteracy", "Income",
"Frost")])
```

- Use `cor()` to get a correlation matrix

```
cor(states)
```

```
            Murder Population Illiteracy Income Frost
Murder        1.00       0.34       0.70  -0.23 -0.54
Population    0.34       1.00       0.11   0.21 -0.33
Illiteracy    0.70       0.11       1.00  -0.44 -0.67
Income       -0.23       0.21      -0.44   1.00  0.23
Frost        -0.54      -0.33      -0.67   0.23  1.00
```

Regression Models in R

- `cor()` produces a correlation matrix but doesn't tell us anything about the statistical significance of the estimates
- `cor.test()` calculates the correlation and tests for significance but only works on a pair at a time
- Years ago I created a function called `Mat.cor.test` that does what `cor.test` does and works with a matrix

Regression Models in R

```
##  matrix cor.test
Mat.cor.test <- function(obj, alt = "two.sided", meth = "pearson")
{
  n <- dim(obj)[2]
  est <- matrix(0, nrow = n, ncol = n,
                    dimnames = list(names(obj), names(obj)))
  pval <- matrix(0, nrow = n, ncol = n,
                    dimnames = list(names(obj), names(obj)))
  for(i in 1:(n - 1)) {
    for(j in (i + 1):n) {
      temp.cor <- cor.test(obj[, i], obj[, j], alternative = alt,
                           method = meth)
      est[i, j] <- temp.cor$estimate
      pval[i, j] <- temp.cor$p.value
      est[j, i] <- temp.cor$estimate
      pval[j, i] <- temp.cor$p.value
    }
  }
  diag(est) = 1
  list(estimates = round(est, 2), p.values = zapsmall(round(pval, 2)))
}
```

Regression Models in R

```
Mat.cor.test(states)
$estimates
           Murder Population Illiteracy Income Frost
Murder       1.00       0.34       0.70  -0.23 -0.54
Population   0.34       1.00       0.11   0.21 -0.33
Illiteracy   0.70       0.11       1.00  -0.44 -0.67
Income      -0.23       0.21      -0.44   1.00  0.23
Frost       -0.54      -0.33      -0.67   0.23  1.00

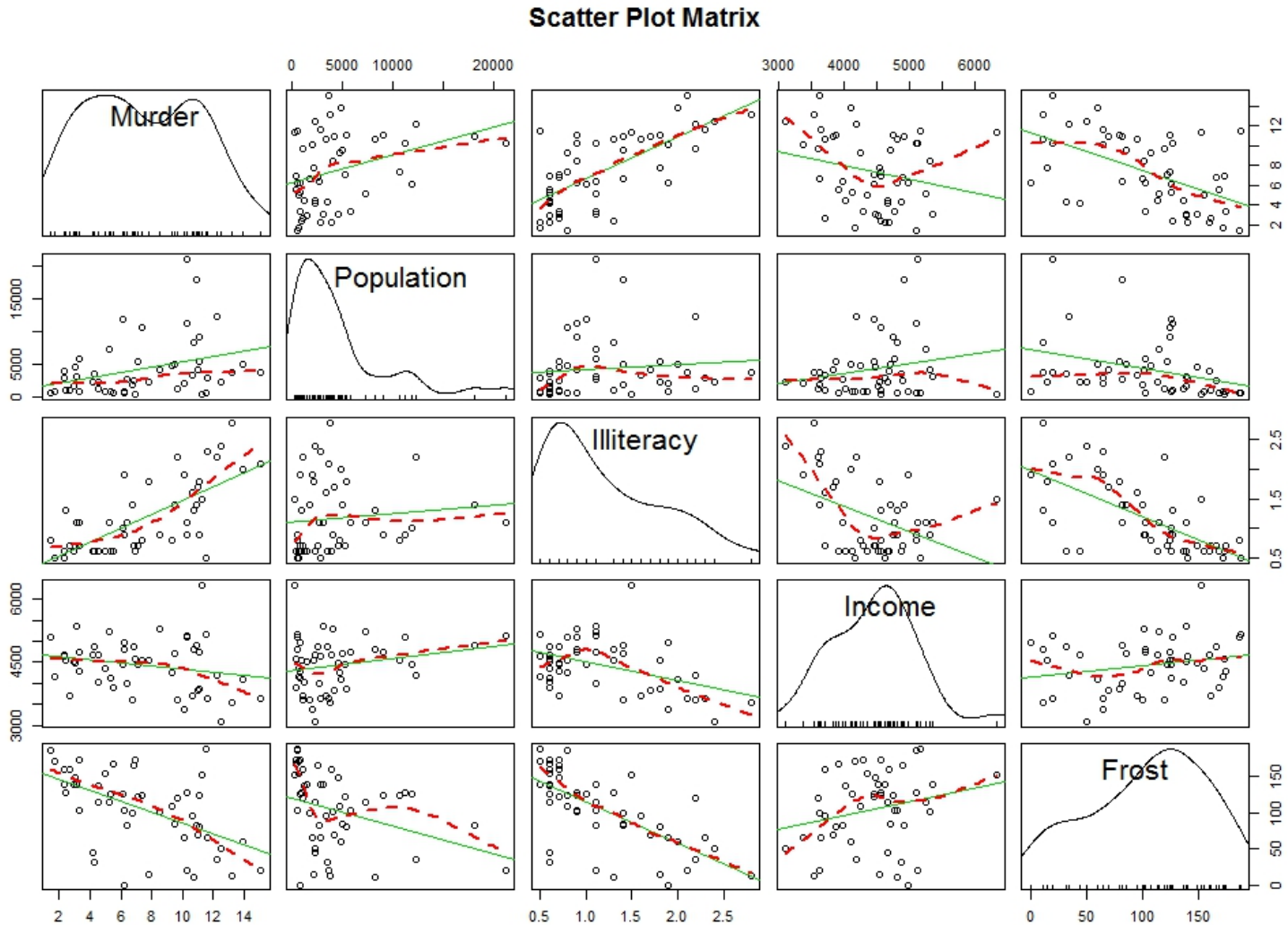
$p.values
           Murder Population Illiteracy Income Frost
Murder       0.00       0.01       0.00   0.11  0.00
Population   0.01       0.00       0.46   0.15  0.02
Illiteracy   0.00       0.46       0.00   0.00  0.00
Income       0.11       0.15       0.00   0.00  0.11
Frost        0.00       0.02       0.00   0.11  0.00
```

Regression Models in R

- It is important to visualize the data
- Let's turn to the `car` library again

```
library(car)
scatterplotMatrix(states, spread=FALSE,
                  smoother.args=list(lty=2),
                   main="Scatter Plot Matrix")
```

Regression Models in R



Regression Models in R

- Fit a multiple regression using the `states` data frame, with `Murder` as the dependent variable...
- ```
  fit <- lm(Murder ~ Population + Illiteracy
  +
          Income + Frost, data=states)
  ```
- `summary(fit)`
- `anova(fit)`
- This model shows use the main effects of each of the variables on the murder rate
- We should look at the interaction between variables when we fit regression models
  ```

Regression Models in R

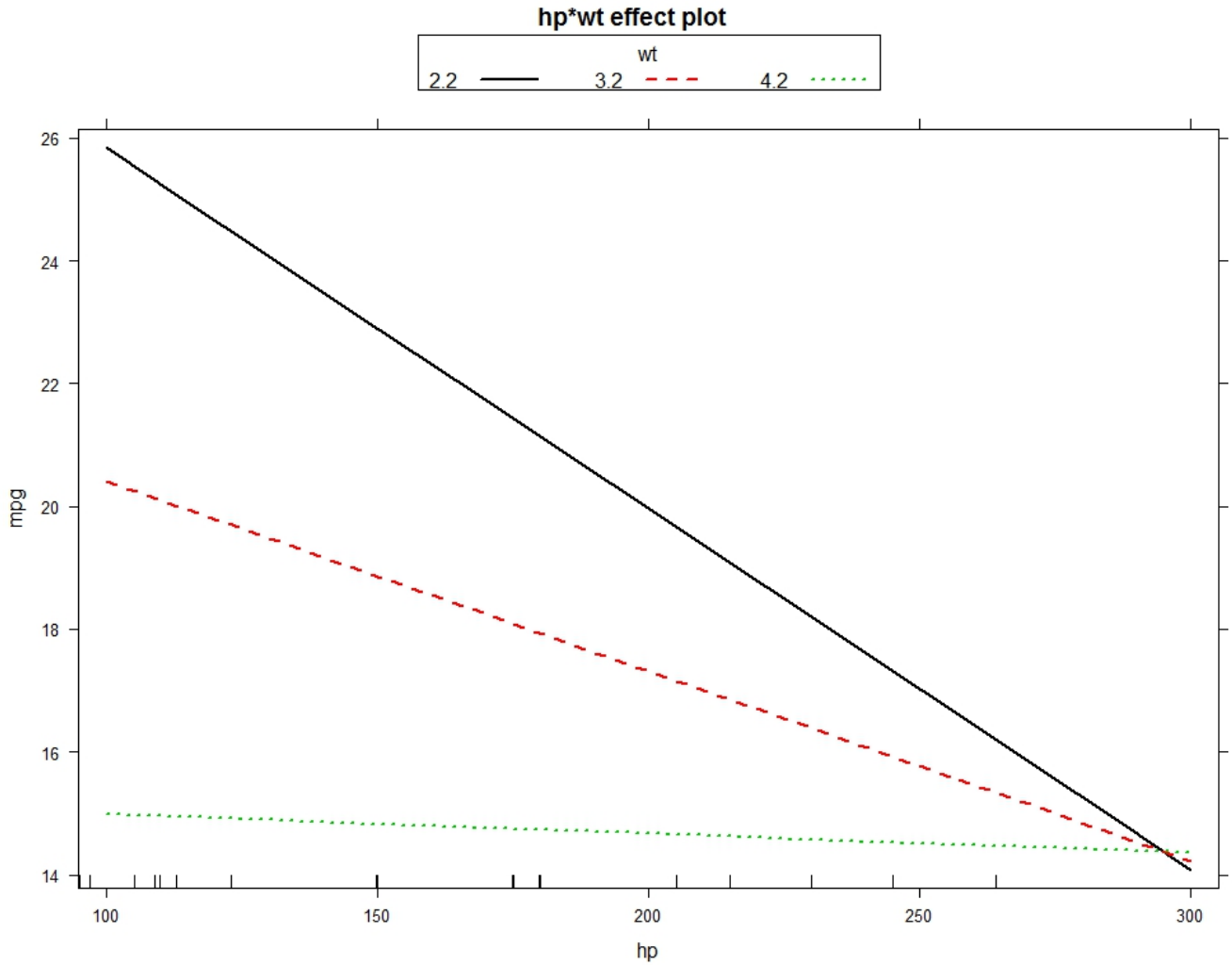
- Let's look at another model

```
fit <- lm(mpg   hp + wt + hp:wt,
data=mtcars)
```

- When we fit a model with interactions we can visualize the interaction terms to learn more about the relationship
- We can use the effects package to help out

```
library(effects)
plot(effect("hp:wt", fit,,
       list(wt=c(2.2,3.2,4.2))),
       multiline=TRUE)
```

Regression Models in R



Regression Models in R

After fitting a regression model it's important to look at diagnostics to check the assumptions

1. *Normality*: the residuals should be normally distributed with a mean of 0 and variance σ^2
2. *Independence*: the dependent variable values are independent
3. *Linearity*: the dependent variable is linearly related to the independent variables
4. *Homoscedasticity*: the variance of the residuals is constant and does not vary with the dependent variable

We also look for outliers, high-leverage observations, and influential observations (*Cook's distance*)

Regression Models in R

Let's return to our earlier model of regessing weight on height

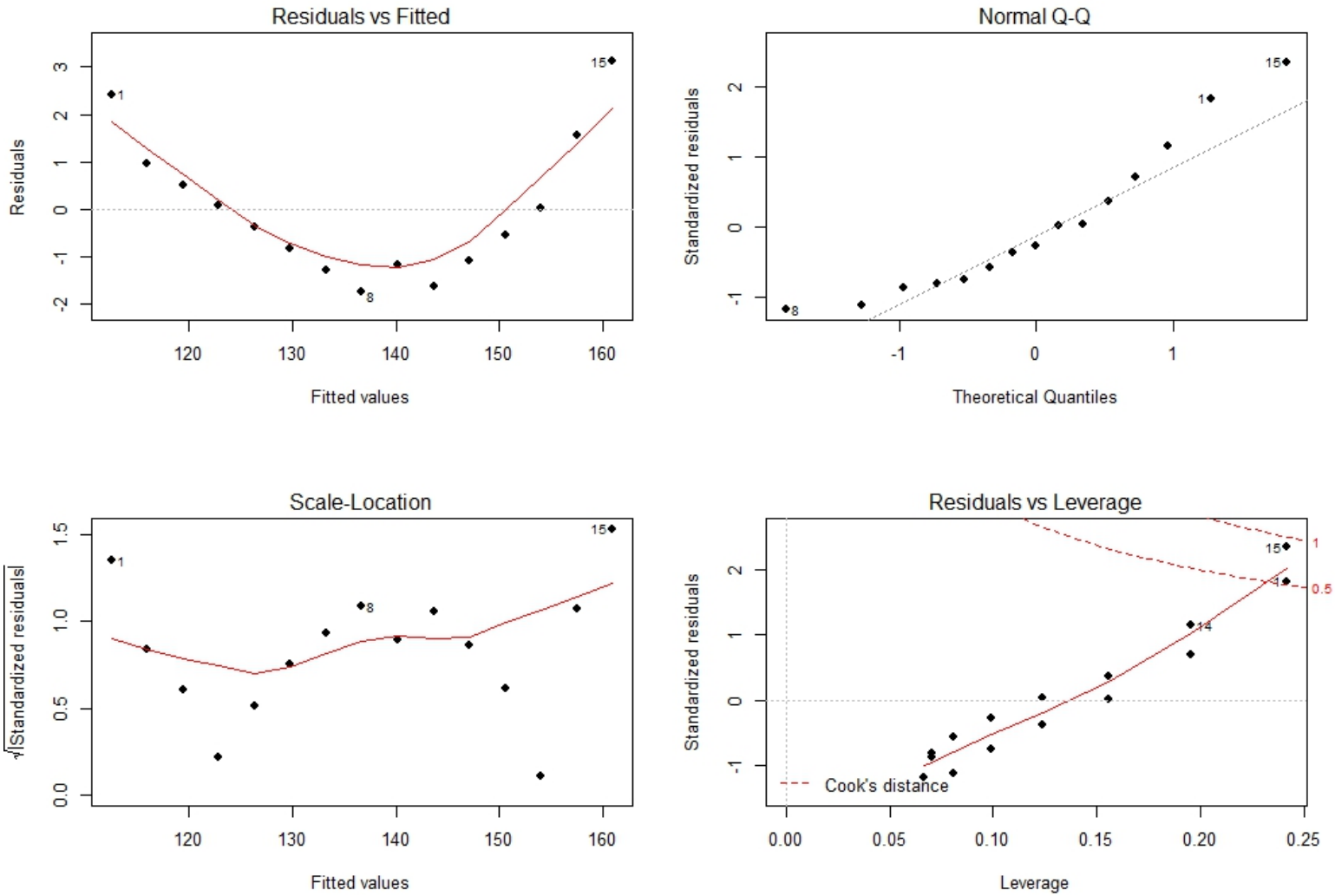
```
fit <- lm(weight ~ height, data=women)

par(mfrow=c(2,2))

plot(fit)

par(mfrow=c(1,1)) ## return parameters to
normal
```

Regression Models in R



Regression Models in R

Look again at the `states` model

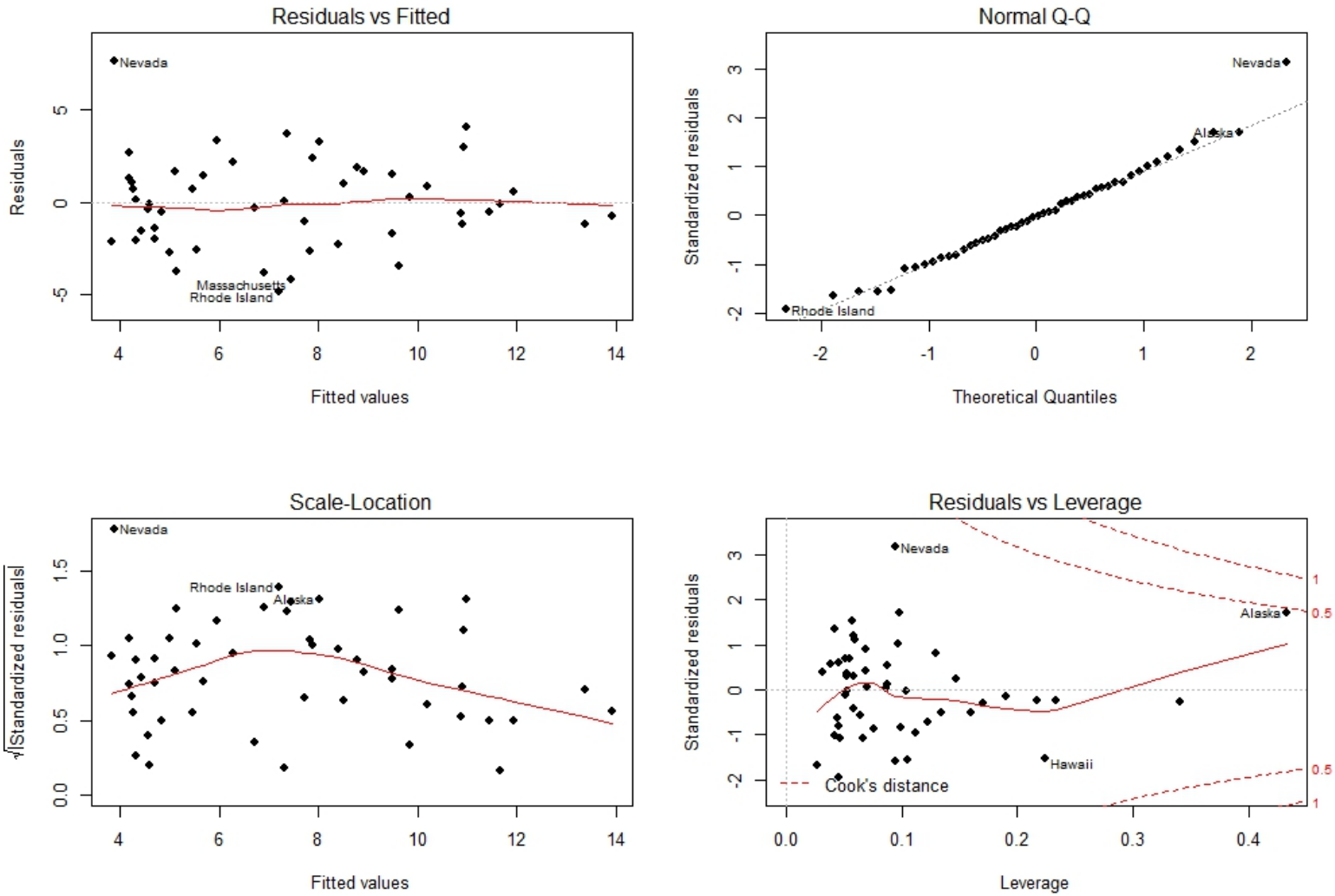
```
fit <- lm(Murder  Population + Illiteracy +
Income + Frost, data=states)

par(mfrow=c(2,2))

plot(fit)

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

Regression Models in R



Regression Models in R

The `car` package provides several additional functions that enhance model evaluation

Function	Purpose
`qqPlot()`	Quantile comparisons plot
`durbinWatsonTest()`	Durbin–Watson test for autocorrelated errors
`crPlots()`	Component plus residual plots
`ncvTest()`	Score test for nonconstant error variance
`spreadLevelPlot()`	Spread-level plots
`outlierTest()`	Bonferroni outlier test
`avPlots()`	Added variable plots
`influencePlot()`	Regression influence plots
`scatterplot()`	Enhanced scatter plots
`scatterplotMatrix()`	Enhanced scatter plot matrixes
`vif()`	Variance inflation factors

Regression Models in R

In-class practice

1. Fit a multiple regression model using the `Auto` data used before
2. If you need to download the data again...
   ```
   AutoLink <-
   "http://www-bcf.usc.edu/ gareth/ISL/Auto.csv"
   ```
3. Make `origin` a factor and be sure `horsepower` is numeric
4. Regress `mpg` on `cylinders, displacement, horsepower, weight, acceleration` and `origin`
5. Change your plot to produce four graphs, two by two
6. Revise the model based on the output from summary()
7. Interpret your final model

Regression Models in R

- Multicollinearity is when two or more independent variables are highly correlated
- Results in large confidence intervals for the parameter estimates (uncertainty)
- Makes interpretation of of the parameters difficult
- May show variables are not significant when they are
- May switch the sign of some parameters
- Variance Inflation Factor (VIF) is a measure to detect multicollinearity – `vif()` in the `car` library

Regression Models in R

After fitting a regression model and interpreting the output, it may be necessary to make revisions

- Delete observations
- Transform variables
- Add or delete variables
- Try another regression modeling approach

Regression Models in R

Finding the "best" model

- Comparing models using `anova` – based on the F for a reduced model vs. a full model

```
> anova(fit.02b, fit.02)
Analysis of Variance Table

Model 1: Murder ~ Population + Illiteracy
Model 2: Murder ~ Population + Illiteracy + Income + Frost
  Res.Df   RSS Df Sum of Sq     F Pr(>F)
1     47 289.2
2     45 289.2  2   0.07851 0.006  0.994
```

- and with `AIC`

```
> AIC(fit.02b, fit.02)
         df     AIC
fit.02b  4 237.657
fit.02   6 241.643
```

Regression Models in R

- Variable selection using step-wise regression, and all subsets regression
- Assessing how well a model works in the real world
  - Cross-validation
    - A portion of the data is selected as the training sample, and a portion is selected as the hold-out sample
    - A regression equation is developed on the training sample and then applied to the hold-out sample
    - The performance on this sample is a more accurate estimate of the operating characteristics of the model with new data
  - Relative Importance
    - Which variables are most important in predicting the outcome?
    - Standardize your data before fitting the regression
    - The coefficients measure how many standard deviations your dependent variable changes with a 1 standard deviation of your independent variable
    - Puts the effects of each variable on the same footing, and comparable

Regression Models in R

- In R we use the `glm()` function to fit generalized linear models
- Two popular models in this framework are logistic regression (dependent variable is categorical) and Poisson regression (dependent variable is a count variable)

Regression Models in R

- In the linear regression framework we model the expected value of Y given $X_1, X_2, \ldots, X_p$

$$\mu_Y = \beta_0 + \sum_{i=1}^{p} \beta_i X_i$$

- In the generalized linear model framework we model a function of the expected value of Y

$$g(\mu_Y) = \beta_0 + \sum_{i=1}^{p} \beta_i X_i$$

where $g(\mu_Y)$ is called a *link function*

Regression Models in R

- The format for the `glm()` is as follows:

  `glm(`*`formula`*`, family=`*`family`*`(link=`*`function`*`), data=)`

  where *family* is the probability distribution, and *function* is the link function

Family	Default link function
binomial	(link = "logit")
gaussian	(link = "identity")
gamma	(link = "inverse")
inverse.gaussian	(link = "1/mu^2")
poisson	(link = "log")
quasi	(link = "identity", variance = "constant")
quasibinomial	(link = "logit")
quasipoisson	(link = "log")

Regression Models in R

- Logistic regression applies to situations in which the response variable is dichotomous (0 or 1)
- The model assumes that Y follows a binomial distribution
- It takes the form

$$\log\left(\frac{\pi}{1-\pi}\right) = \beta_0 + \sum_{i=1}^{p} \beta_i X_i$$

where $\pi = \mu_Y$ is the conditional mean of Y (that is, the probability that $Y = 1$ given a set of X values), $\left(\frac{\pi}{1-\pi}\right)$ is the *odds* that $Y = 1$, and $\log\left(\frac{\pi}{1-\pi}\right)$ is the log odds, or *logit*

Regression Models in R

- $\log\left(\frac{\pi}{1-\pi}\right)$ is the link function
- The probability distribution is binomial
- Suppose we have a data frame called `mydata` with dependent variable Y and independent variables $X_1, X_2,$ and X_3
- We can fit our logistic regression model as follows:

```
glm(Y~X1+X2+X3, family=binomial(link="logit"),
data=mydata)
```

Regression Models in R

Let's consider an example..

```
install.packages("AER")
data(Affairs, package = "AER")
summary(Affairs)
table(Affairs$affairs)
```

Looking at the table we see `Affairs$affairs` contains count data
– we can convert count data to a binary variable for our logistic regression

```
## create a dichotomous variable from the counts
Affairs$ynaffair[Affairs$affairs > 0] <- 1
Affairs$ynaffair[Affairs$affairs == 0] <- 0
```

Regression Models in R

```
Affairs$ynaffair <- factor(Affairs$ynaffair,
levels=c(0,1),
labels=c("No","Yes"))
table(Affairs$ynaffair)
```

Now our dependent variable is in a form we can use for logistic regression

Regression Models in R

Fit a logistic regression model using all the variables

```
fit.full <- glm(ynaffair ~ gender + age
+ yearsmarried + children
+ religiousness + education
+ occupation +rating,
data=Affairs, family=binomial())
summary(fit.full)
```

What do you observe?

Regression Models in R

Fit a reduced logistic regression model using only the significant variables

```
fit.reduced <- glm(ynaffair ~ age
+ yearsmarried + religiousness
+ rating, data=Affairs,
family=binomial())
summary(fit.reduced)
```

What do you observe?

Use `anova()` to compare nested models

```
anova(fit.reduced, fit.full, test="Chisq")
```

Regression Models in R

Let's try to interpret the model

```
coef(fit.reduced)
(Intercept)     age yearsmarried religiousness rating
      1.931 -0.035         0.101          -0.329 -0.461
```

Since the logistic output is log odds the coefficients are difficult to interpret

Let's look at the odds instead...

```
exp(coef(fit.reduced))
(Intercept)   age yearsmarried religiousness rating
      6.895 0.965         1.106          0.720  0.630
```

Regression Models in R

Here's what we observe

```
exp(coef(fit.reduced))
(Intercept)   age yearsmarried religiousness rating
      6.895 0.965        1.106          0.720  0.630
```

- For each year increase in `yearsmarried` the odds of having an affair increase by a factor of 1.106
- The odds decrease for a unit increase in each of `age`, `religiousness`, and `rating`
- Since none of the variables takes on a value of 0 the intercept has no interpretation

For the effect of an n unit change we use $(e^{\hat{\beta}_j})^n$

Regression Models in R

What is often most useful is the probability of the "outcome" so we transform the output to a probability

$$\Pr(Y = 1|\mathbf{X}) = \frac{e^{\hat{\beta}_0 + \sum \hat{\beta}_i X_i}}{1 + e^{\hat{\beta}_0 + \sum \hat{\beta}_i X_i}}$$

in R we just use the `predict()` function

Regression Models in R

Let's use our fitted model to test the effect of `rating` on the probability of having an affair

```
testdata <- data.frame(rating=c(1, 2, 3, 4, 5),
age=mean(Affairs$age),
yearsmarried=mean(Affairs$yearsmarried),
religiousness=mean(Affairs$religiousness))

testdata$prob <- predict(fit.reduced,
newdata=testdata,
type="response")

testdata
```

Regression Models in R

Now let's use our fitted model to test the effect of `age` on the probability of having an affair

```
testdata <- data.frame(rating=mean(Affairs$rating),
age=seq(17, 57, 10),
yearsmarried=mean(Affairs$yearsmarried),
religiousness=mean(Affairs$religiousness))

testdata$prob <- predict(fit.reduced,
newdata=testdata,
type="response")

testdata
```

Regression Models in R

- Recall that in the logistic model Y is assumed to have a binomial distribution
- We validate this assumption by testing for *overdispersion*
- Overdispersion occurs when the observed variance of the response variable is larger than what would be expected from a binomial distribution; this can lead to inaccurate tests of significance

Regression Models in R

- One way to detect overdispersion is to compare the residual deviance to the residual degrees of freedom in the model

$$\phi = \frac{Residual\ Deviance}{Resdiual\ df}$$

if $\phi >> 1$ there is evidence of overdispersion

```
summary(fit.reduced)$deviance/df.residual(fit.reduced)
```

Regression Models in R

- Or fit the same model but use `family=quasibinomial()` instead of `family=binomial()`

```
fit.od <- glm(ynaffair ~ age + yearsmarried
+ religiousness + rating,
family = quasibinomial(),
data = Affairs)

pchisq(summary(fit.od)$dispersion * fit.full$df.residual,
fit.full$df.residual, lower = F)
```

Regression Models in R

- While using the results of a logistic regression model to estimate the probability or *likelihood* of the event there are other useful interpretations
- We saw earlier that the (*conditional*) probability is the odds ratio divided by one plus the odds ratio
- Relative risk is the ratio of the conditional probability of one group to another
- Relative risk allows us to compare the risk between two groups
- Using our earlier result we see the probability of a 17 year old having an extra-marital affair is 0.335; for a 57 year old it is 0.109
- $$RR = 0.335/0.109 \approx 3.07$$

  a 17 year old is about 3 times more likely to have an extra-marital affair than a 57 year old

Regression Models in R

- Probabilities estimated from logistic regression models are point estimates $\hat{\pi}$
- We can compute a confidence interval for the probability
- First we need the standard error of the probability estimate

$$SE(\hat{\pi}) = \hat{\pi}(1 - \hat{\pi})SE(logit)$$

- A 95% confidence interval is

$$\hat{\pi} \pm 1.96 \times SE(\hat{\pi})$$

Regression Models in R

Extensions of Logistic Regression

- *Robust logistic regression* –The `glmRob()` function in the `robust` package can be used to fit a robust logistic regression model helpful when fitting logistic regression models to data containing outliers and influential observations
- *Multinomial logistic regression* –When the response variable has more than two unordered categories (for example, arried/widowed/divorced), you can fit a polytomous logistic regression using the `mlogit()` function in the `mlogit` package
- *Ordinal logistic regression* – When the response variable is a set of ordered categories (for example, credit risk as poor/good/excellent), you can fit an ordinal logistic regression using the `lrm()` function in the `rms` package

Regression Models in R

Recap of Logistic Regression

- Logistic regression is a generalized linear model used for modeling binary, multi-level, and ordered dependent variables
- The transformation for a logistic regression converts the binary outcomes to the log of the odds ratio
- We can transform the predicted values into a probability
- Logistic regression is good first technique for modeling probabilities
- When we set a threshold probability value we can use a logistic regression model as a classifier
- If the Predicted probability exceeds the threshold we classify the observation as *true*, otherwise it is *false*
- We use the `glm()` function in R to fit a logistic regression

Poisson Regression Models

- In the example using the `Affairs` data frame we converted the number of affairs to a binary variable
- If we wanted to predict the number of affairs rather than the probability of having an affair we would fit a Poisson regression

```
data(Affairs, package = "AER")
pois.fit <- glm(affairs ~ gender+age+yearsmarrie
children+religiousness+education+
occupation+rating, data=Affairs,
family = poisson())
summary(pois.fit)
```

The Poisson regression has the form

$$\log(Y) = \sum_{i=0}^{p} \beta_i X_i$$

Poisson Regression Models

- We update the model to remove the insignificant variables

```
pois.fit2 <- update(pois.fit, .~.-gender-childre
-education)
summary(pois.fit2)
```

- Let's use our Poisson model to predict the number of affairs

```
newaffairs <- data.frame(age=c(22,32,42),
yearsmarried=mean(Affairs$yearsmarried),
religiousness=mean(Affairs$religiousness),
occupation=mean(Affairs$occupation),
rating=mean(Affairs$rating))
predict(pois.fit2, newdata = newaffairs, type="response")
```

Poisson Regression Models – Summary

- Poisson regression is useful when the dependent variable represents count data
- The transformed dependent variable is $\log(Y)$
- When predicting from a Poisson model remember to use `type=''response''` or exponentiate the raw output

Missing Values

- Sample data sets used in teaching data analysis tend to be complete
- Real world data are often incomplete and we need to deal with it appropriately
- We'll explore how to identify and deal with missing data so we can get the most actionable information form our data

Missing Values

- When dealing with missing data we need to identify the missing data
- Find the causes of the missing data
- Deal with the issue
  - Delete the cases with missing data or
  - Repalce the missing values with reasonable data values

Missing Values

Classifying Missing Data

Missing Completely at Random (MCAR) The data are MCAR if the presence of missing data on a variable is unrelated to any other observed or unobserved variable; there is no systematic reason why the data are missing

Missing at Random (MAR) The data are MAR if the presence of missing data on a variable is related to other observed variables but not to its own unobserved values

Not Missing at Random (NMAR) If the missing data on a variable are neither MCAR nor MAR then the data are NMAR

Missing Values

Table 3.1: R functions for identifying missing values

x	is.na(x)	is.nan(x)	is.infinite(x)
x <- NA	TRUE	FALSE	FALSE
x <- 0/0	TRUE	TRUE	FALSE
x <- 1/0	FALSE	FALSE	TRUE

Missing Values

Other functions to help us understand the missing data

- Load the `sleep` data from the `VIM` package
  ```
  data(sleep, package="VIM")
  ```
- List the rows without missing data using
  ```
  complete.cases()
  sleep[complete.cases(sleep),]
  ```
- List the ones with missing data
  ```
  sleep[!complete.cases(sleep),]
  ```
- Since `TRUE` and `FALSE` are equivalent to 1 and 0 we can do
  ```
  sum(is.na(sleep$Dream))
  mean(is.na(sleep$Dream))
  mean(!complete.cases(sleep))
  ```
- `complete.cases()` returns true for `NA` and `NaN` and not `Inf` and `-Inf`

Missing Values

Exploring missing data for patterns

- While the previous functions help us identify missing data there are other ways to understand missing data
- The `mice` package has a function `md.pattern()` that tabulates missing data

```
library(mice)
data(sleep, package="VIM")
md.pattern(sleep)
```

Missing Values

Visually exploring missing data for patterns

- Besides summarizing the missing data in a table we can visually inspect for patterns of missingness

```
library("VIM")
```

- Plot the number of missing values for each variable alone, and for each combination of variables

```
aggr(sleep, prop=FALSE, numbers=TRUE)
```

- Display missing values for each observation – lighter colors are lower values for the variable and darker are higher – red represents missing values

```
matrixplot(sleep)
```

- Plot the relationship between two variables and their distributions given missing vales

```
marginplot(sleep[c("Gest", "Dream")], pch
=20, col = c("darkgray", "red", "blue"))
```

Missing Values

The questions we want to address are...

- What percentage of the data is missing?
- Are the missing data concentrated in a few variables or widely distributed?
- Do the missing values appear to be random?
- Does the anything suggest a possible mechanism that's producing the missing values?

Missing Values

- There are several approaches we can take with missing data
- Rationalize the missing value from other variables
  - Suppose we have a survey and we want to group respondents by birth year
  - We collect date of birth, and age
  - If date of birth is missing we can fill-in the birth year from age and today's date
- Complete case analysis which means delete any observation (row) that is missing one or more value – only analyze complete data
- Use multiple imputation to impute the missing values

Missing Values

- Multiple imputation (MI) provides an approach to missing values based on repeated simulations
- MI is a widely-used method for complex missing-values problems
- Typically a set of 3 to 10 complete datasets is generated from an existing dataset that's missing values
- Monte Carlo methods are used to fill in the missing data in each of the simulated datasets
- Finally standard statistical methods are applied to each of the simulated datasets and the outcomes are combined
- These provide estimated results and confidence intervals that take into account the uncertainty introduced by the missing values

Missing Values

```
fit.na <- lm(Dream ~ Span + Gest, data =
sleep)
summary(fit.na)
```

```
library(mice)
imp <- mice(sleep, m=5)
fit.mi <- with(imp, lm(Dream ~ Span + Gest))
pooled <- pool(fit.mi)
summary(pooled)
```

- We see from the results that the pooled estimate is close to the original using mising data, but the pooled results provide additional information

Methods for Missing Data – Summary

- When working with real data we will almost always encounter missing values
- To deal with missing values in your data you can delete the entire observation (complete cases) or find a suitable value to use
- Missing data may be MCAR, MAR, NMAR
- If the data are missing at random and a small percentage of your data then deletion is OK
- Impute a value through rationalization
- Multiple imputation provides an approach based on simulations

References

James, G., Hastie, T., Witten, D. and Tibshirani, R. (2013).
An Introduction to Statistical Learning with Applications in R.
Springer, second edition.
8th Printing 2017.

Kabacoff, R. I. (2015).
R in Action.
Manning, Shelter Island, NY, second edition.

Lander, J. P. (2014).
R for Everyone.
Addison-Wesley, Upper Saddle River.

Zumel, N. and Mount, J. (2014).
Practical Data Science with R.
Manning, Shelter Island, NY, second edition.