Chapter 1

R statistical software and review

The purpose of this chapter is to discuss R in the context of a quick review of the topics we covered last semester in ADA1[1].

1.1 R

R is a programming language for programming, data management, and statistical analysis. So many people have written "An Introduction to R", that I refer you to the course website[2] for links to tutorials. I encourage you to learn R by (1) running the commands in the tutorials, (2) looking at the help for the commands (e.g., `?mean`), and (3) trying things on your own as you become curious. Make mistakes, figure out why some things don't work the way you expect, and keep trying. Persistence wins the day with programming (as does asking and searching for help).

R is more difficult to master (though, more rewarding) than some statistical packages (such as Minitab) for the following reasons: (1) R does not, in general, provide a point-and-click environment for statistical analysis. Rather,

[1] `http://statacumen.com/teaching/ada1/`

[2] `http://statacumen.com/teaching/ada2/`

R uses syntax-based programs (i.e., code) to define, transform, and read data, and to define the procedures for analyzing data. (2) R does not really have a spreadsheet environment for data management. Rather, data are entered directly within an R program, read from a file, or imported from a spreadsheet. All manipulation, transformation, and selection of data is coded in the R program. Well done, this means that all the steps of the analysis are available to be repeatable and understood.

Take a minute to install the packages we'll need this semester by executing the following commands in R.

```
#### Install packages needed this semester
ADA2.package.list <- c("BSDA",       "Hmisc",    "MASS",          "NbClust",
                       "aod",        "candisc",  "car",           "cluster",
                       "ellipse",    "ggplot2",  "gridExtra",     "klaR",
                       "leaps",      "lsmeans",  "moments",       "multcomp",
                       "mvnormtest", "nortest",  "plyr",          "popbio",
                       "reshape",    "reshape2", "scatterplot3d", "vcd",
                       "vioplot",    "xtable")
install.packages(ADA2.package.list)
```

1.2 ADA1 Ch 0: R warm-up

This example illustrates several strategies for data summarization and analysis. The data for this example are from 15 turkeys raised on farms in either Virginia or Wisconsin.

You should use **help** to get more information on the functions demonstrated here. Many of the lines in the program have comments, which helps anyone reading the program to more easily follow the logic. I **strongly recommend** commenting any line of code that isn't absolutely obvious what is being done and why it is being done. I also recommend placing comments before blocks of code in order to describe what the code below is *meant* to do.

```
#### Example: Turkey, R warm-up
# Read the data file from the website and learn some functions

# filename
fn.data <- "http://statacumen.com/teach/ADA2/ADA2_notes_Ch01_turkey.csv"
# read file and assign data to turkey variable
turkey <- read.csv(fn.data)
```

```
# examine the structure of the dataset, is it what you expected?
  # a data.frame containing integers, numbers, and factors
str(turkey)
```

```
## 'data.frame': 15 obs. of  3 variables:
##  $ age   : int  28 20 32 25 23 22 29 27 28 26 ...
##  $ weight: num  13.3 8.9 15.1 13.8 13.1 10.4 13.1 12.4 13.2 11.8 ...
##  $ orig  : Factor w/ 2 levels "va","wi": 1 1 1 2 2 1 1 1 1 1 ...
```

```
# print dataset to screen
turkey
```

```
##    age weight orig
## 1   28   13.3   va
## 2   20    8.9   va
## 3   32   15.1   va
## 4   25   13.8   wi
## 5   23   13.1   wi
## 6   22   10.4   va
## 7   29   13.1   va
## 8   27   12.4   va
## 9   28   13.2   va
## 10  26   11.8   va
## 11  21   11.5   wi
## 12  31   16.6   wi
## 13  27   14.2   wi
## 14  29   15.4   wi
## 15  30   15.9   wi
```

```
# Note: to view the age variable (column), there's a few ways to do that
turkey$age          # name the variable
```

```
##  [1] 28 20 32 25 23 22 29 27 28 26 21 31 27 29 30
```

```
turkey[, 1]         # give the column number
```

```
##  [1] 28 20 32 25 23 22 29 27 28 26 21 31 27 29 30
```

```
turkey[, "age"]     # give the column name
```

```
##  [1] 28 20 32 25 23 22 29 27 28 26 21 31 27 29 30
```

```
# and the structure is a vector
str(turkey$age)
```

```
##  int [1:15] 28 20 32 25 23 22 29 27 28 26 ...
```

```
# let's create an additional variable for later
# gt25mo will be a variable indicating whether the age is greater than 25 months
turkey$gt25mo <- (turkey$age > 25)
# now we also have a Boolean (logical) column
str(turkey)
```

```
## 'data.frame': 15 obs. of  4 variables:
##  $ age   : int  28 20 32 25 23 22 29 27 28 26 ...
##  $ weight: num  13.3 8.9 15.1 13.8 13.1 10.4 13.1 12.4 13.2 11.8 ...
##  $ orig  : Factor w/ 2 levels "va","wi": 1 1 1 2 2 1 1 1 1 1 ...
##  $ gt25mo: logi  TRUE FALSE TRUE FALSE FALSE FALSE ...
```

```
# there are a couple ways of subsetting the rows
turkey[(turkey$gt25mo == TRUE),]  # specify the rows
```

```
##    age weight orig gt25mo
## 1   28   13.3   va   TRUE
## 3   32   15.1   va   TRUE
## 7   29   13.1   va   TRUE
## 8   27   12.4   va   TRUE
## 9   28   13.2   va   TRUE
## 10  26   11.8   va   TRUE
## 12  31   16.6   wi   TRUE
## 13  27   14.2   wi   TRUE
## 14  29   15.4   wi   TRUE
## 15  30   15.9   wi   TRUE
```

```
subset(turkey, gt25mo == FALSE)   # use subset() to select the data.frame records
```

```
##    age weight orig gt25mo
## 2   20    8.9   va  FALSE
## 4   25   13.8   wi  FALSE
## 5   23   13.1   wi  FALSE
## 6   22   10.4   va  FALSE
## 11  21   11.5   wi  FALSE
```

Analyses can be then done on the entire dataset, or repeated for all subsets of a variable in the dataset.

```
# summaries of each variable in the entire dataset,
summary(turkey)
```

```
##       age            weight      orig     gt25mo
##  Min.   :20.00   Min.   : 8.90   va:8   Mode :logical
##  1st Qu.:24.00   1st Qu.:12.10   wi:7   FALSE:5
##  Median :27.00   Median :13.20          TRUE :10
##  Mean   :26.53   Mean   :13.25          NA's :0
##  3rd Qu.:29.00   3rd Qu.:14.65
##  Max.   :32.00   Max.   :16.60
```

```
# or summarize by a variable in the dataset.
by(turkey, turkey$orig, summary)
```

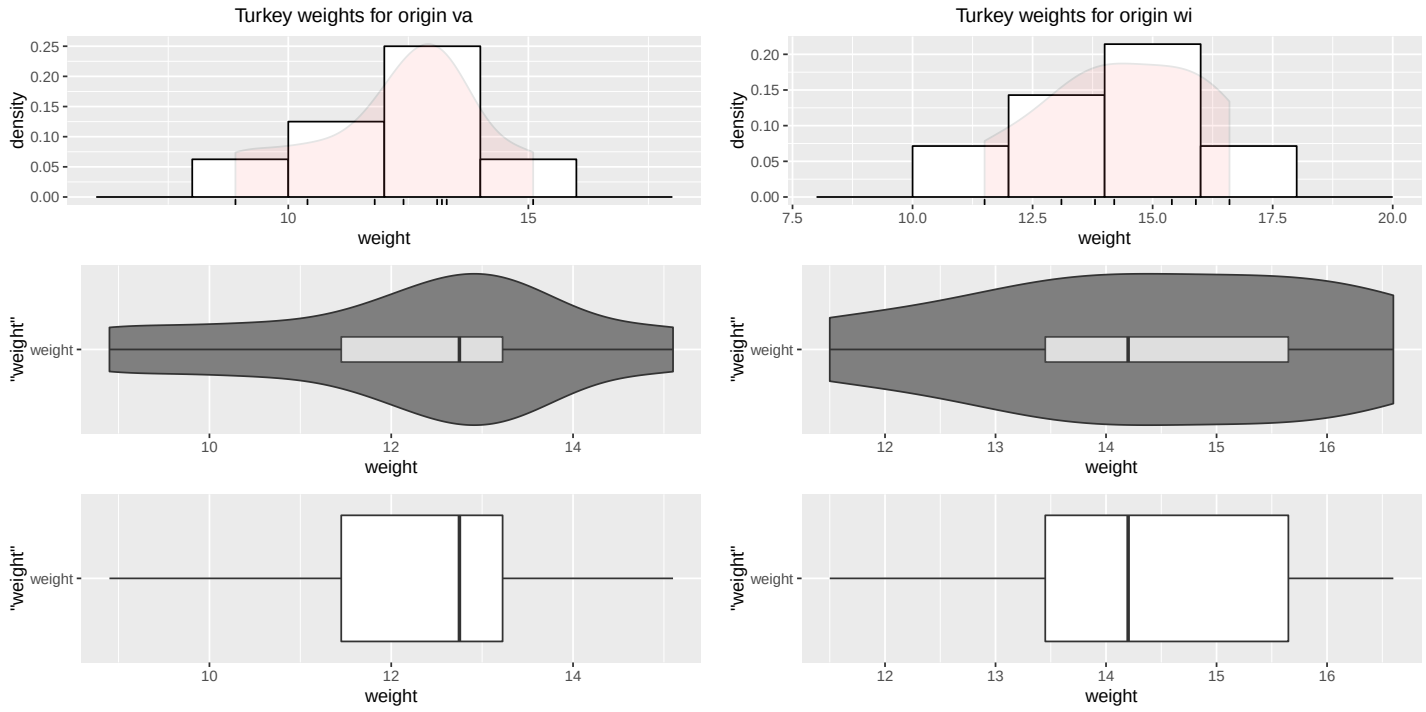
```
## turkey$orig: va
##       age            weight      orig     gt25mo
##  Min.   :20.00   Min.   : 8.90   va:8   Mode :logical
##  1st Qu.:25.00   1st Qu.:11.45   wi:0   FALSE:2
##  Median :27.50   Median :12.75          TRUE :6
##  Mean   :26.50   Mean   :12.28          NA's :0
##  3rd Qu.:28.25   3rd Qu.:13.22
##  Max.   :32.00   Max.   :15.10
## ------------------------------------------------------------
## turkey$orig: wi
##       age            weight      orig     gt25mo
##  Min.   :21.00   Min.   :11.50   va:0   Mode :logical
##  1st Qu.:24.00   1st Qu.:13.45   wi:7   FALSE:3
```

```
##  Median :27.00   Median :14.20          TRUE :4
##  Mean   :26.57   Mean   :14.36          NA's :0
##  3rd Qu.:29.50   3rd Qu.:15.65
##  Max.   :31.00   Max.   :16.60
```

1.3 ADA1 Chapters 2, 4, 6: Estimation in one-sample problems

Plot the weights by origin.

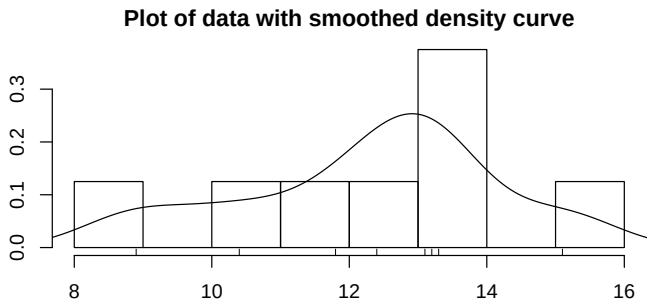
```
#### Example: Turkey, Chapters 2, 4, 6
# subset the data for convenience
turkeyva <- subset(turkey, orig == "va")
turkeywi <- subset(turkey, orig == "wi")

library(ggplot2)
# Histogram overlaid with kernel density curve
p11 <- ggplot(turkeyva, aes(x = weight))
  # Histogram with density instead of count on y-axis
p11 <- p11 + geom_histogram(aes(y=..density..)
                            , binwidth=2
                            , colour="black", fill="white")
  # Overlay with transparent density plot
p11 <- p11 + geom_density(alpha=0.1, fill="#FF6666")
p11 <- p11 + geom_rug()

# violin plot
p12 <- ggplot(turkeyva, aes(x = "weight", y = weight))
p12 <- p12 + geom_violin(fill = "gray50")
p12 <- p12 + geom_boxplot(width = 0.2, alpha = 3/4)
p12 <- p12 + coord_flip()

# boxplot
p13 <- ggplot(turkeyva, aes(x = "weight", y = weight))
p13 <- p13 + geom_boxplot()
p13 <- p13 + coord_flip()

library(gridExtra)
#grid.arrange(p11, p12, p13, ncol=1, main="Turkey weights for origin va")
## add grobs = list(), and main= becomes top=
grid.arrange(grobs = list(p11, p12, p13), ncol=1, top="Turkey weights for origin va")

# Histogram overlaid with kernel density curve
p21 <- ggplot(turkeywi, aes(x = weight))
  # Histogram with density instead of count on y-axis
p21 <- p21 + geom_histogram(aes(y=..density..)
                            , binwidth=2
                            , colour="black", fill="white")
  # Overlay with transparent density plot
p21 <- p21 + geom_density(alpha=0.1, fill="#FF6666")
p21 <- p21 + geom_rug()

# violin plot
p22 <- ggplot(turkeywi, aes(x = "weight", y = weight))
p22 <- p22 + geom_violin(fill = "gray50")
p22 <- p22 + geom_boxplot(width = 0.2, alpha = 3/4)
p22 <- p22 + coord_flip()

# boxplot
p23 <- ggplot(turkeywi, aes(x = "weight", y = weight))
p23 <- p23 + geom_boxplot()
p23 <- p23 + coord_flip()

library(gridExtra)
grid.arrange(grobs = list(p21, p22, p23), ncol=1, top="Turkey weights for origin wi")
```

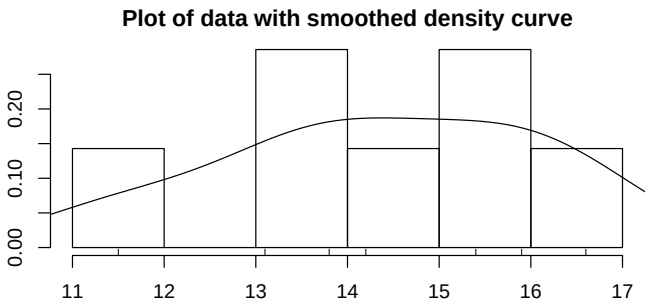


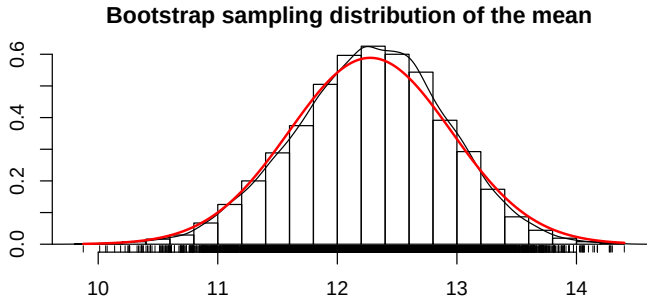
Check normality of each sample graphically with with bootstrap sampling distribution and normal quantile plot and formally with normality tests.

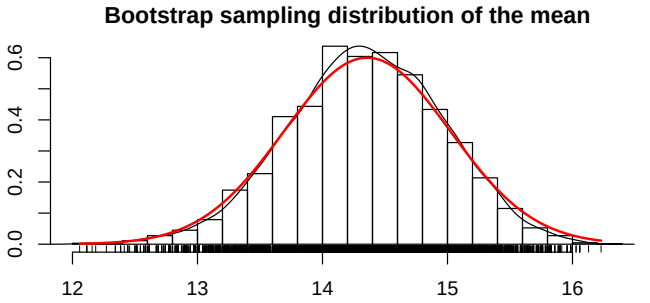
```
# a function to compare the bootstrap sampling distribution with
#   a normal distribution with mean and SEM estimated from the data
bs.one.samp.dist <- function(dat, N = 1e4) {
  n <- length(dat);
  # resample from data
  sam <- matrix(sample(dat, size = N * n, replace = TRUE), ncol=N);
  # draw a histogram of the means
  sam.mean <- colMeans(sam);
  # save par() settings
  old.par <- par(no.readonly = TRUE)
  # make smaller margins
  par(mfrow=c(2,1), mar=c(3,2,2,1), oma=c(1,1,1,1))
  # Histogram overlaid with kernel density curve
  hist(dat, freq = FALSE, breaks = 6
       , main = "Plot of data with smoothed density curve")
  points(density(dat), type = "l")
  rug(dat)

  hist(colMeans(sam), freq = FALSE, breaks = 25
       , main = "Bootstrap sampling distribution of the mean"
       , xlab = paste("Data: n =", n
                      , ", mean =", signif(mean(dat), digits = 5)
                      , ", se =", signif(sd(dat)/sqrt(n)), digits = 5))
  # overlay a density curve for the sample means
  points(density(sam.mean), type = "l")
  # overlay a normal distribution, bold and red
  x <- seq(min(sam.mean), max(sam.mean), length = 1000)
  points(x, dnorm(x, mean = mean(dat), sd = sd(dat)/sqrt(n))
         , type = "l", lwd = 2, col = "red")
  # place a rug of points under the plot
  rug(sam.mean)
  # restore par() settings
  par(old.par)
}
```

```
# Bootstrap sampling distribution
bs.one.samp.dist(turkeyva$weight)
bs.one.samp.dist(turkeywi$weight)
```

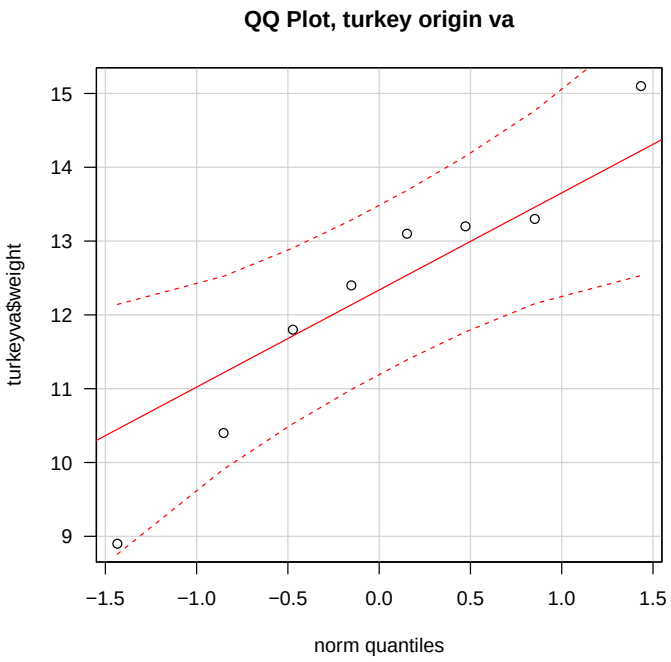


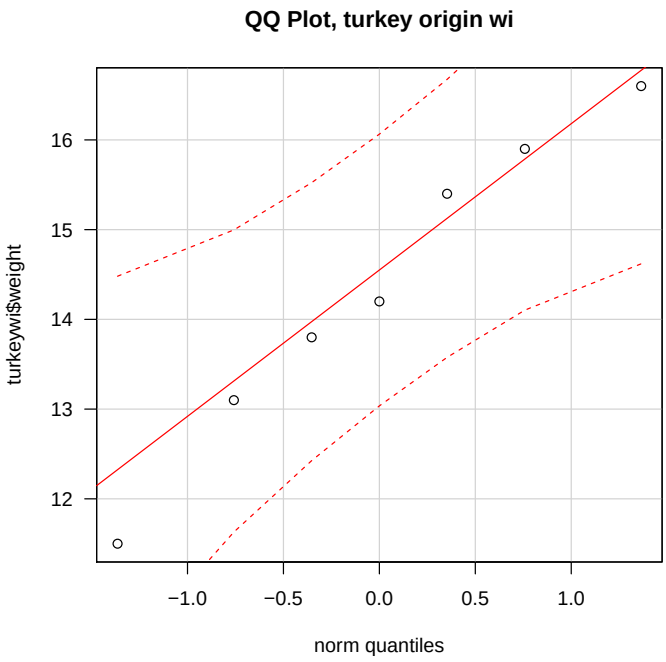







```
# normal quantile-quantile (QQ) plot of each orig sample
library(car)
# qq plot
#   las    =  1 : turns labels on y-axis to read horizontally
#   id.n   =  n : labels n most extreme observations, and outputs to console
#   id.cex =  1 : is the size of those labels
#   lwd    =  1 : line width
qqPlot(turkeyva$weight, las = 1, id.n = 0, id.cex = 1, lwd = 1
     , main="QQ Plot, turkey origin va")
qqPlot(turkeywi$weight, las = 1, id.n = 0, id.cex = 1, lwd = 1
     , main="QQ Plot, turkey origin wi")
```






```
# Normality tests

# VA
shapiro.test(turkeyva$weight)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  turkeyva$weight
## W = 0.95414, p-value = 0.7528
```

```
library(nortest)
ad.test(turkeyva$weight)
```

```
##
##  Anderson-Darling normality test
##
## data:  turkeyva$weight
## A = 0.283, p-value = 0.5339
```

```
# lillie.test(turkeyva£weight)
cvm.test(turkeyva$weight)
```

```
##
##  Cramer-von Mises normality test
##
## data:  turkeyva$weight
## W = 0.050135, p-value = 0.4642
```

```
# WI
shapiro.test(turkeywi$weight)
```

```
##
##  Shapiro-Wilk normality test
```

```
##
## data:  turkeywi$weight
## W = 0.97326, p-value = 0.9209
```

```
library(nortest)
ad.test(turkeywi$weight)
```

```
## Error in ad.test(turkeywi$weight):  sample size must be greater than 7
# lillie.test(turkeywi$weight)
cvm.test(turkeywi$weight)
## Error in cvm.test(turkeywi$weight):  sample size must be greater than 7
## Note: The errors above are expected.
```

Because we do not have any serious departures from normality (the data are consistent with being normal, as well as the sampling distribution of the mean) the t-test is appropriate. We will also look at a couple nonparametric methods.

```
# Is the average turkey weight 12 lbs?

# t-tests of the mean

# VA
t.summary <- t.test(turkeyva$weight, mu = 12)
t.summary
```

```
##
##  One Sample t-test
##
## data:  turkeyva$weight
## t = 0.40582, df = 7, p-value = 0.697
## alternative hypothesis: true mean is not equal to 12
## 95 percent confidence interval:
##  10.67264 13.87736
## sample estimates:
## mean of x
##    12.275
```

```
# WI
t.summary <- t.test(turkeywi$weight, mu = 12)
t.summary
```

```
##
##  One Sample t-test
##
## data:  turkeywi$weight
## t = 3.5442, df = 6, p-value = 0.01216
## alternative hypothesis: true mean is not equal to 12
## 95 percent confidence interval:
##  12.72978 15.98450
## sample estimates:
## mean of x
##  14.35714
```

```r
# Sign test for the median

# VA
library(BSDA)
SIGN.test(turkeyva$weight, md=12)
```

```
##
##  One-sample Sign-Test
##
## data:  turkeyva$weight
## s = 5, p-value = 0.7266
## alternative hypothesis: true median is not equal to 12
## 95 percent confidence interval:
##   9.9125 13.8850
## sample estimates:
## median of x
##       12.75
##                   Conf.Level  L.E.pt U.E.pt
## Lower Achieved CI     0.9297 10.4000 13.300
## Interpolated CI       0.9500  9.9125 13.885
## Upper Achieved CI     0.9922  8.9000 15.100
```

```r
# WI
SIGN.test(turkeywi$weight, md=12)
```

```
##
##  One-sample Sign-Test
##
## data:  turkeywi$weight
## s = 6, p-value = 0.125
## alternative hypothesis: true median is not equal to 12
## 95 percent confidence interval:
##  12.00286 16.38000
## sample estimates:
## median of x
##        14.2
##                   Conf.Level  L.E.pt U.E.pt
## Lower Achieved CI     0.8750 13.1000  15.90
## Interpolated CI       0.9500 12.0029  16.38
## Upper Achieved CI     0.9844 11.5000  16.60
```

```r
# Wilcoxon sign-rank test for the median (or mean, since symmetric assumption)

# VA
# with continuity correction in the normal approximation for the p-value
wilcox.test(turkeyva$weight, mu=12, conf.int=TRUE)
```

```
## Warning in wilcox.test.default(turkeyva$weight, mu = 12, conf.int = TRUE): cannot compute
exact p-value with ties
## Warning in wilcox.test.default(turkeyva$weight, mu = 12, conf.int = TRUE): cannot compute
exact confidence interval with ties
##
```

```
##  Wilcoxon signed rank test with continuity correction
##
## data:  turkeyva$weight
## V = 21.5, p-value = 0.674
## alternative hypothesis: true location is not equal to 12
## 95 percent confidence interval:
##  10.40005 14.09997
## sample estimates:
## (pseudo)median
##       12.47331
```

```
# without continuity correction
wilcox.test(turkeyva$weight, mu=12, conf.int=TRUE, correct=FALSE)
```

```
## Warning in wilcox.test.default(turkeyva$weight, mu = 12, conf.int = TRUE, :  cannot compute
exact p-value with ties
## Warning in wilcox.test.default(turkeyva$weight, mu = 12, conf.int = TRUE, :  cannot compute
exact confidence interval with ties
##
##  Wilcoxon signed rank test
##
## data:  turkeyva$weight
## V = 21.5, p-value = 0.6236
## alternative hypothesis: true location is not equal to 12
## 95 percent confidence interval:
##  10.64999 13.75002
## sample estimates:
## (pseudo)median
##       12.47331
```

```
# WI
# with continuity correction in the normal approximation for the p-value
wilcox.test(turkeywi$weight, mu=12, conf.int=TRUE)
```

```
##
##  Wilcoxon signed rank test
##
## data:  turkeywi$weight
## V = 27, p-value = 0.03125
## alternative hypothesis: true location is not equal to 12
## 95 percent confidence interval:
##  12.65 16.00
## sample estimates:
## (pseudo)median
##         14.375
```

```
# without continuity correction
wilcox.test(turkeywi$weight, mu=12, conf.int=TRUE, correct=FALSE)
```

```
##
##  Wilcoxon signed rank test
##
## data:  turkeywi$weight
```

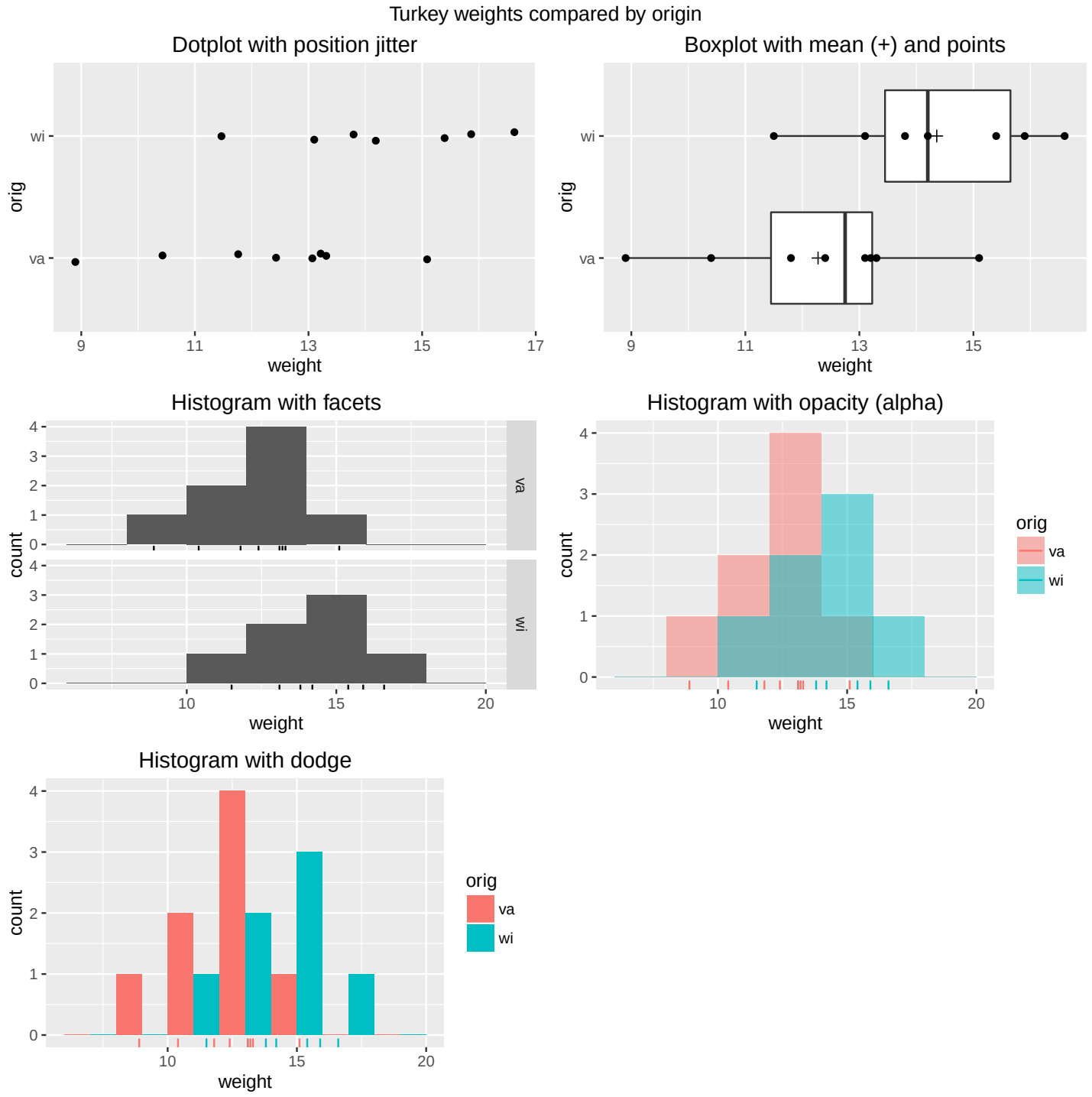
```
## V = 27, p-value = 0.03125
## alternative hypothesis: true location is not equal to 12
## 95 percent confidence interval:
##  12.65 16.00
## sample estimates:
## (pseudo)median
##         14.375
```

1.4 ADA1 Chapters 3, 4, 6: Two-sample inferences

Presume it is of interest to compare the center of the weight distributions between the origins. There are many ways to plot the data for visual comparisons.

```
#### Example: Turkey, Chapters 3, 4, 6
# stripchart (dotplot) using ggplot
library(ggplot2)
p1 <- ggplot(turkey, aes(x = weight, y = orig))
p1 <- p1 + geom_point(position = position_jitter(h=0.1))
p1 <- p1 + labs(title = "Dotplot with position jitter")

# boxplot
p2 <- ggplot(turkey, aes(x = orig, y = weight))
p2 <- p2 + geom_boxplot()
# add a "+" at the mean
p2 <- p2 + stat_summary(fun.y = mean, geom = "point", shape = 3, size = 2)
p2 <- p2 + geom_point()
p2 <- p2 + coord_flip()
p2 <- p2 + labs(title = "Boxplot with mean (+) and points")

# histogram using ggplot
p3 <- ggplot(turkey, aes(x = weight))
p3 <- p3 + geom_histogram(binwidth = 2)
p3 <- p3 + geom_rug()
p3 <- p3 + facet_grid(orig ~ .)
p3 <- p3 + labs(title = "Histogram with facets")

p4 <- ggplot(turkey, aes(x = weight, fill=orig))
p4 <- p4 + geom_histogram(binwidth = 2, alpha = 0.5, position="identity")
p4 <- p4 + geom_rug(aes(colour = orig))
p4 <- p4 + labs(title = "Histogram with opacity (alpha)")

p5 <- ggplot(turkey, aes(x = weight, fill=orig))
p5 <- p5 + geom_histogram(binwidth = 2, alpha = 1, position="dodge")
```

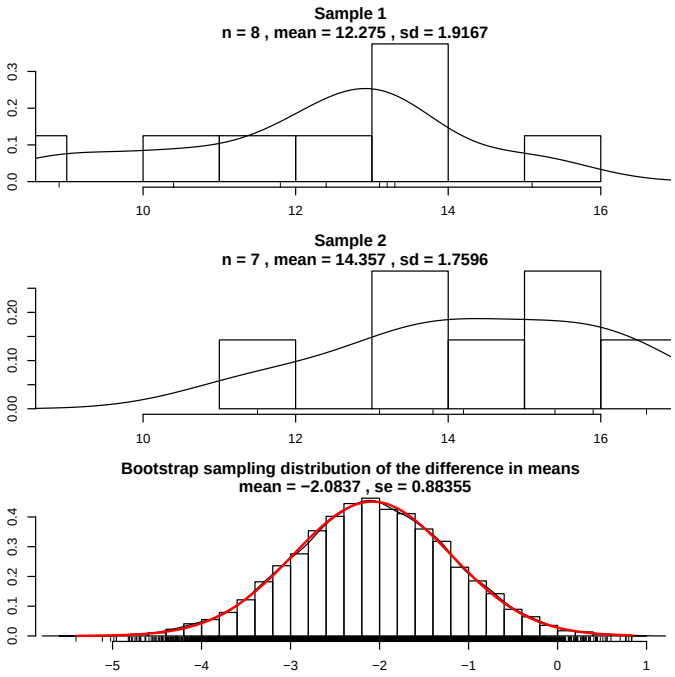
```
p5 <- p5 + geom_rug(aes(colour = orig))
p5 <- p5 + labs(title = "Histogram with dodge")

library(gridExtra)
grid.arrange(grobs = list(p1, p2, p3, p4, p5), ncol=2, nrow=3
             , top="Turkey weights compared by origin")
```



Using the two-sample t-test, first check the normality assumptions of the sampling distribution of the mean difference between the populations.

```
# a function to compare the bootstrap sampling distribution
#   of the difference of means from two samples with
#   a normal distribution with mean and SEM estimated from the data
bs.two.samp.diff.dist <- function(dat1, dat2, N = 1e4) {
  n1 <- length(dat1);
  n2 <- length(dat2);
  # resample from data
  sam1 <- matrix(sample(dat1, size = N * n1, replace = TRUE), ncol=N);
  sam2 <- matrix(sample(dat2, size = N * n2, replace = TRUE), ncol=N);
  # calculate the means and take difference between populations
  sam1.mean <- colMeans(sam1);
  sam2.mean <- colMeans(sam2);
  diff.mean <- sam1.mean - sam2.mean;
  # save par() settings
  old.par <- par(no.readonly = TRUE)
  # make smaller margins
  par(mfrow=c(3,1), mar=c(3,2,2,1), oma=c(1,1,1,1))
  # Histogram overlaid with kernel density curve
  hist(dat1, freq = FALSE, breaks = 6
      , main = paste("Sample 1", "\n"
                     , "n =", n1
                     , ", mean =", signif(mean(dat1), digits = 5)
                     , ", sd =", signif(sd(dat1), digits = 5))
      , xlim = range(c(dat1, dat2)))
  points(density(dat1), type = "l")
  rug(dat1)

  hist(dat2, freq = FALSE, breaks = 6
      , main = paste("Sample 2", "\n"
                     , "n =", n2
                     , ", mean =", signif(mean(dat2), digits = 5)
                     , ", sd =", signif(sd(dat2), digits = 5))
      , xlim = range(c(dat1, dat2)))
  points(density(dat2), type = "l")
  rug(dat2)

  hist(diff.mean, freq = FALSE, breaks = 25
      , main = paste("Bootstrap sampling distribution of the difference in means", "\n"
                   , "mean =", signif(mean(diff.mean), digits = 5)
                   , ", se =", signif(sd(diff.mean), digits = 5)))
  # overlay a density curve for the sample means
  points(density(diff.mean), type = "l")
  # overlay a normal distribution, bold and red
  x <- seq(min(diff.mean), max(diff.mean), length = 1000)
  points(x, dnorm(x, mean = mean(diff.mean), sd = sd(diff.mean))
       , type = "l", lwd = 2, col = "red")
  # place a rug of points under the plot
  rug(diff.mean)
  # restore par() settings
  par(old.par)
}
```

```
bs.two.samp.diff.dist(turkeyva$weight, turkeywi$weight)
```



Two-sample t-test is appropriate since the bootstrap sampling distribution of the difference in means is approximately normal. This is the most powerful test and detects a difference at a 0.05 significance level.

```
# Two-sample t-test
## Equal variances
#   var.equal = FALSE is the default
# two-sample t-test specifying two separate vectors
t.summary.eqvar <- t.test(turkeyva$weight, turkeywi$weight, var.equal = TRUE)
t.summary.eqvar
```

```
##
##  Two Sample t-test
##
## data:  turkeyva$weight and turkeywi$weight
## t = -2.1796, df = 13, p-value = 0.04827
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -4.14595971 -0.01832601
## sample estimates:
## mean of x mean of y
##  12.27500  14.35714
```

```
# two-sample t-test with unequal variances (Welch = Satterthwaite)
#   specified using data.frame and a formula, HeadBreadth by Group
t.summary.uneqvar <- t.test(weight ~ orig, data = turkey, var.equal = FALSE)
t.summary.uneqvar
```

```
##
##  Welch Two Sample t-test
##
## data:  weight by orig
```

```
## t = -2.1929, df = 12.956, p-value = 0.04717
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -4.13407696 -0.03020875
## sample estimates:
## mean in group va mean in group wi
##         12.27500         14.35714
```

(Wilcoxon-)Mann-Whitney two-sample test is appropriate because the shapes of the two distributions are similar, though their locations are different. This is a less powerful test, but doesn't require normality, and fails to detect a difference at a 0.05 significance level.

```
# with continuity correction in the normal approximation for the p-value
wilcox.test(turkeyva$weight, turkeywi$weight, conf.int=TRUE)
```

```
## Warning in wilcox.test.default(turkeyva$weight, turkeywi$weight, conf.int = TRUE): cannot
compute exact p-value with ties
## Warning in wilcox.test.default(turkeyva$weight, turkeywi$weight, conf.int = TRUE): cannot
compute exact confidence intervals with ties
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  turkeyva$weight and turkeywi$weight
## W = 11.5, p-value = 0.06384
## alternative hypothesis: true location shift is not equal to 0
## 95 percent confidence interval:
##  -4.19994493  0.09993686
## sample estimates:
## difference in location
##              -2.152771
```

```
# without continuity correction
wilcox.test(turkeyva$weight, turkeywi$weight, conf.int=TRUE, correct=FALSE)
```

```
## Warning in wilcox.test.default(turkeyva$weight, turkeywi$weight, conf.int = TRUE, :  cannot
compute exact p-value with ties
## Warning in wilcox.test.default(turkeyva$weight, turkeywi$weight, conf.int = TRUE, :  cannot
compute exact confidence intervals with ties
##
##  Wilcoxon rank sum test
##
## data:  turkeyva$weight and turkeywi$weight
## W = 11.5, p-value = 0.05598
## alternative hypothesis: true location shift is not equal to 0
## 95 percent confidence interval:
##  -4.100049e+00  1.445586e-05
## sample estimates:
## difference in location
##              -2.152771
```

1.5 ADA1 Chapters 5, 4, 6: One-way ANOVA

The Waste Run-up data[3] refer to five suppliers of the Levi-Strauss clothing manufacturing plant in Albuquerque. The firm's quality control department collects weekly data on percent-age waste (run-up) relative to what can be achieved by computer layouts of patterns on cloth. It is possible to have negative values, which indicate that the plant employees beat the computer in controlling waste. Under question are differences among the five supplier plants (PT1, . . . , PT5).

```
#### Example: Waste Run-up, Chapters 5, 4, 6
# convert to a data.frame by reading the text table
waste <- read.table(text = "
 PT1   PT2   PT3   PT4   PT5
 1.2  16.4  12.1  11.5  24.0
10.1  -6.0   9.7  10.2  -3.7
-2.0 -11.6   7.4   3.8   8.2
 1.5  -1.3  -2.1   8.3   9.2
-3.0   4.0  10.1   6.6  -9.3
-0.7  17.0   4.7  10.2   8.0
 3.2   3.8   4.6   8.8  15.8
 2.7   4.3   3.9   2.7  22.3
-3.2  10.4   3.6   5.1   3.1
-1.7   4.2   9.6  11.2  16.8
 2.4   8.5   9.8   5.9  11.3
 0.3   6.3   6.5  13.0  12.3
 3.5   9.0   5.7   6.8  16.9
-0.8   7.1   5.1  14.5    NA
19.4   4.3   3.4   5.2    NA
 2.8  19.7  -0.8   7.3    NA
13.0   3.0  -3.9   7.1    NA
42.7   7.6   0.9   3.4    NA
 1.4  70.2   1.5   0.7    NA
 3.0   8.5    NA    NA    NA
 2.4   6.0    NA    NA    NA
 1.3   2.9    NA    NA    NA
", header=TRUE)
waste
```

```
##     PT1   PT2  PT3  PT4  PT5
## 1   1.2  16.4 12.1 11.5 24.0
```

[3]From `http://lib.stat.cmu.edu/DASL/Stories/wasterunup.html`, the Data and Story Library (DASL, pronounced "dazzle") is an online library of datafiles and stories that illustrate the use of basic statistics methods. "Waste Run-up" dataset from L. Koopmans, Introduction to Contemporary Statistical Methods, Duxbury Press, 1987, p. 86.

```
## 2  10.1  -6.0  9.7 10.2 -3.7
## 3  -2.0 -11.6  7.4  3.8  8.2
## 4   1.5  -1.3 -2.1  8.3  9.2
## 5  -3.0   4.0 10.1  6.6 -9.3
## 6  -0.7  17.0  4.7 10.2  8.0
## 7   3.2   3.8  4.6  8.8 15.8
## 8   2.7   4.3  3.9  2.7 22.3
## 9  -3.2  10.4  3.6  5.1  3.1
## 10 -1.7   4.2  9.6 11.2 16.8
## 11  2.4   8.5  9.8  5.9 11.3
## 12  0.3   6.3  6.5 13.0 12.3
## 13  3.5   9.0  5.7  6.8 16.9
## 14 -0.8   7.1  5.1 14.5   NA
## 15 19.4   4.3  3.4  5.2   NA
## 16  2.8  19.7 -0.8  7.3   NA
## 17 13.0   3.0 -3.9  7.1   NA
## 18 42.7   7.6  0.9  3.4   NA
## 19  1.4  70.2  1.5  0.7   NA
## 20  3.0   8.5   NA   NA   NA
## 21  2.4   6.0   NA   NA   NA
## 22  1.3   2.9   NA   NA   NA
```

```
library(reshape2)
waste.long <- melt(waste,
              # id.vars: ID variables
              #   all variables to keep but not split apart on
              # id.vars=NULL,
              # measure.vars: The source columns
              #   (if unspecified then all other variables are measure.vars)
              # measure.vars = c("PT1","PT2","PT3","PT4","PT5"),
              # variable.name: Name of the destination column identifying each
              #   original column that the measurement came from
              variable.name = "plant",
              # value.name: column name for values in table
              value.name = "runup",
              # remove the NA values
              na.rm = TRUE
            )
```

```
## No id variables; using all as measure variables
str(waste.long)
```

```
## 'data.frame': 95 obs. of  2 variables:
##  $ plant: Factor w/ 5 levels "PT1","PT2","PT3",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ runup: num  1.2 10.1 -2 1.5 -3 -0.7 3.2 2.7 -3.2 -1.7 ...
```

```
head(waste.long)
```

```
##   plant runup
## 1   PT1   1.2
## 2   PT1  10.1
## 3   PT1  -2.0
```

```
## 4   PT1   1.5
## 5   PT1  -3.0
## 6   PT1  -0.7
```

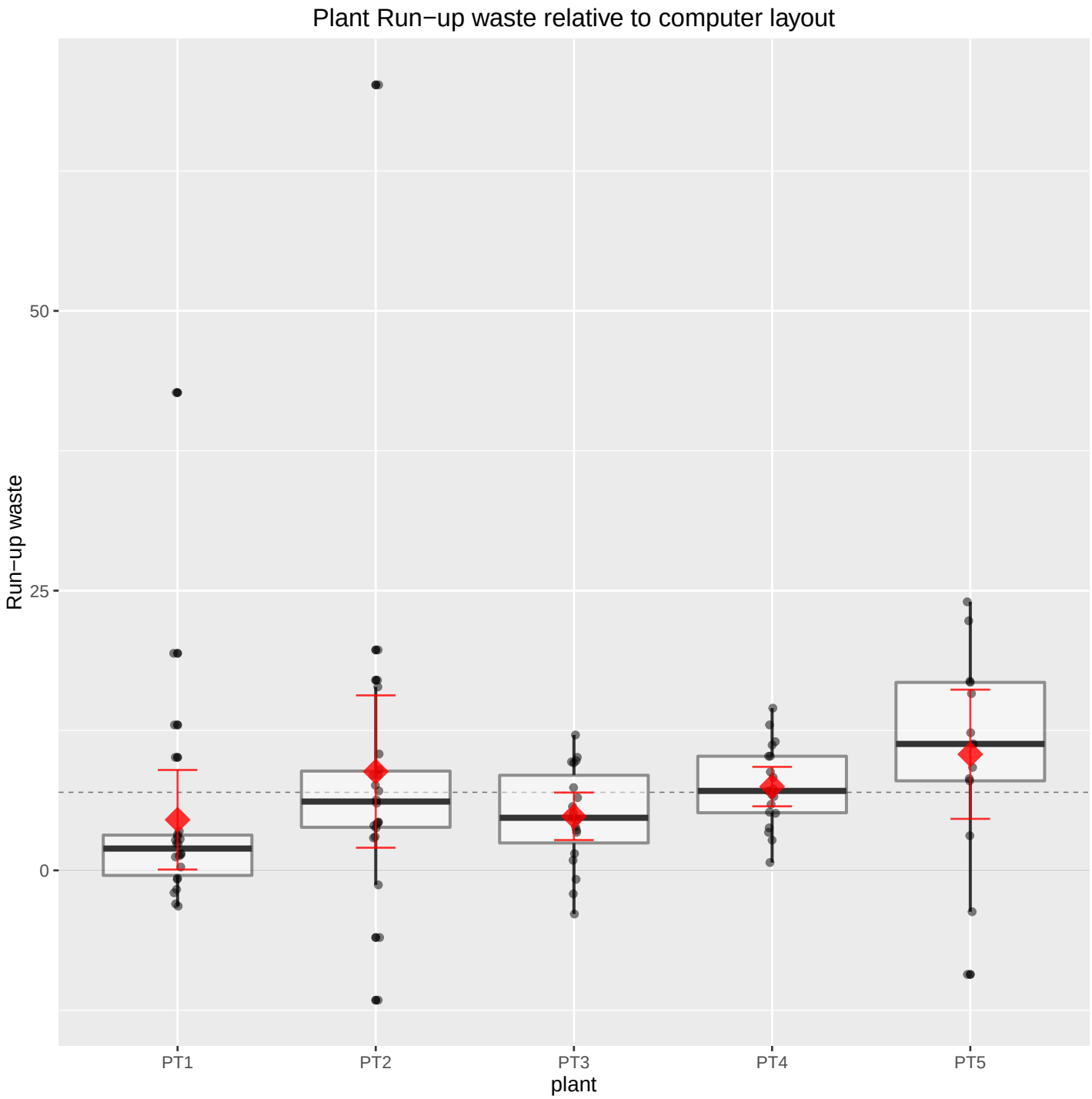
```
tail(waste.long)
```

```
##     plant runup
## 96    PT5  22.3
## 97    PT5   3.1
## 98    PT5  16.8
## 99    PT5  11.3
## 100   PT5  12.3
## 101   PT5  16.9
```

```
# Calculate the mean, sd, n, and se for the plants

# The plyr package is an advanced way to apply a function to subsets of data
#   "Tools for splitting, applying and combining data"
library(plyr)
# ddply "dd" means the input and output are both data.frames
waste.summary <- ddply(waste.long,
                       "plant",
                       function(X) {
                         data.frame( m = mean(X$runup),
                                     s = sd(X$runup),
                                     n = length(X$runup)
                                   )
                       }
                     )
# standard errors
waste.summary$se  <- waste.summary$s/sqrt(waste.summary$n)
waste.summary$moe <- qt(1 - 0.05 / 2, df = waste.summary$n - 1) * waste.summary$se
# individual confidence limits
waste.summary$ci.l <- waste.summary$m - waste.summary$moe

waste.summary$ci.u <- waste.summary$m + waste.summary$moe
waste.summary
```

```
##   plant         m         s  n        se      moe       ci.l
## 1   PT1  4.522727 10.032041 22 2.1388383 4.447958 0.07476963
## 2   PT2  8.831818 15.353467 22 3.2733701 6.807346 2.02447241
## 3   PT3  4.831579  4.403162 19 1.0101547 2.122256 2.70932276
## 4   PT4  7.489474  3.657093 19 0.8389946 1.762662 5.72681139
## 5   PT5 10.376923  9.555030 13 2.6500884 5.774047 4.60287651
##        ci.u
## 1  8.970685
## 2 15.639164
## 3  6.953835
## 4  9.252136
## 5 16.150970
```

```
# Plot the data using ggplot
library(ggplot2)
p <- ggplot(waste.long, aes(x = plant, y = runup))
# plot a reference line for the global mean (assuming no groups)
p <- p + geom_hline(aes(yintercept = 0),
                    colour = "black", linetype = "solid", size = 0.2, alpha = 0.3)
p <- p + geom_hline(aes(yintercept = mean(runup)),
                    colour = "black", linetype = "dashed", size = 0.3, alpha = 0.5)
# boxplot, size=.75 to stand out behind CI
p <- p + geom_boxplot(size = 0.75, alpha = 0.5)
# points for observed data
p <- p + geom_point(position = position_jitter(w = 0.05, h = 0), alpha = 0.5)
# diamond at mean for each group
p <- p + stat_summary(fun.y = mean, geom = "point", shape = 18, size = 6,
                      colour = "red", alpha = 0.8)
# confidence limits based on normal distribution
p <- p + stat_summary(fun.data = "mean_cl_normal", geom = "errorbar",
                      width = .2, colour = "red", alpha = 0.8)
p <- p + labs(title = "Plant Run-up waste relative to computer layout")
p <- p + ylab("Run-up waste")
print(p)
```



The outliers here suggest the ANOVA is not an appropriate model. The normality tests below suggest the distributions for the first two plants are not normal.

```
by(waste.long$runup, waste.long$plant, ad.test)
## waste.long$plant: PT1
##
##  Anderson-Darling normality test
##
## data:  dd[x, ]
```

```
## A = 2.8685, p-value = 1.761e-07
##
## ------------------------------------------------------
## waste.long$plant: PT2
##
##  Anderson-Darling normality test
##
## data:  dd[x, ]
## A = 2.5207, p-value = 1.334e-06
##
## ------------------------------------------------------
## waste.long$plant: PT3
##
##  Anderson-Darling normality test
##
## data:  dd[x, ]
## A = 0.23385, p-value = 0.7624
##
## ------------------------------------------------------
## waste.long$plant: PT4
##
##  Anderson-Darling normality test
##
## data:  dd[x, ]
## A = 0.12363, p-value = 0.9834
##
## ------------------------------------------------------
## waste.long$plant: PT5
##
##  Anderson-Darling normality test
##
## data:  dd[x, ]
## A = 0.27445, p-value = 0.6004
```

For review purposes, I'll fit the ANOVA, but we would count on the following nonparametric method for inference.

```
fit.w <- aov(runup ~ plant, data = waste.long)
summary(fit.w)
```

```
##             Df Sum Sq Mean Sq F value Pr(>F)
## plant        4    451  112.73    1.16  0.334
## Residuals   90   8749   97.21
```

```
fit.w
```

```
## Call:
##    aov(formula = runup ~ plant, data = waste.long)
##
## Terms:
##                     plant Residuals
## Sum of Squares   450.921  8749.088
```

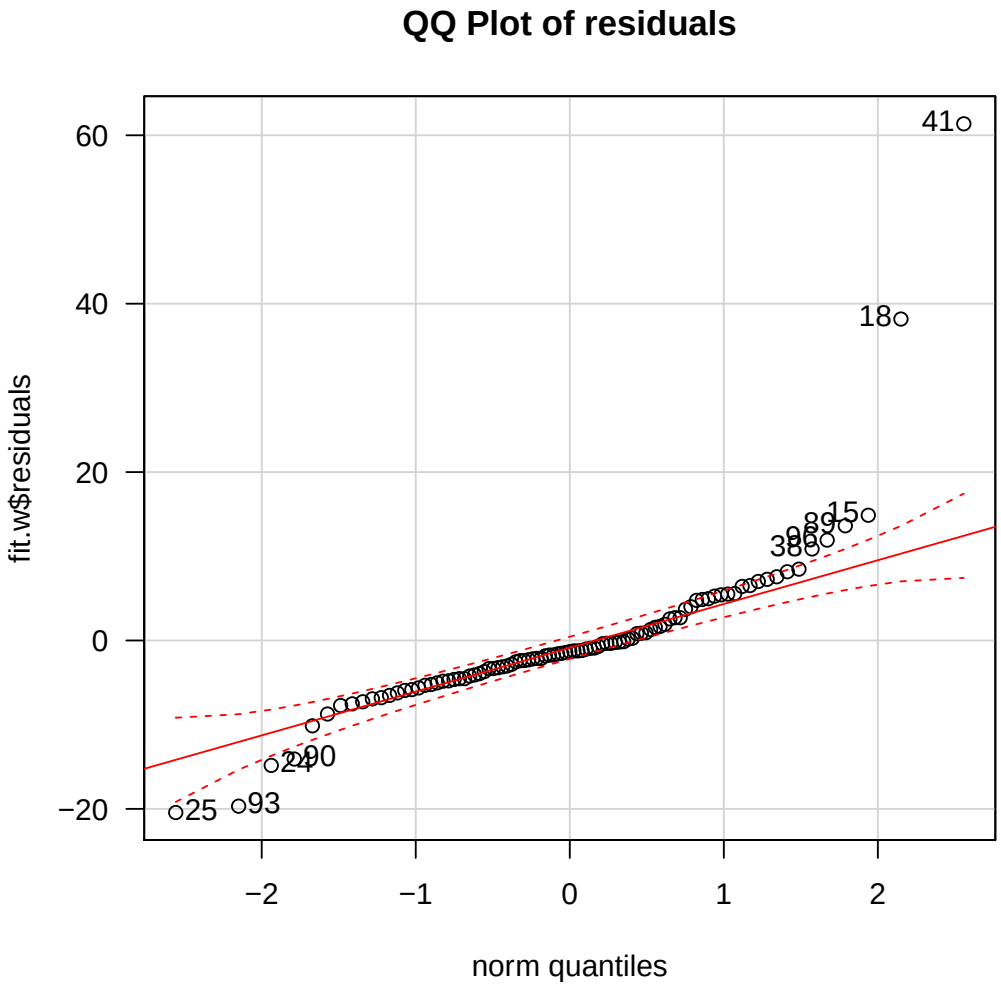
```
## Deg. of Freedom        4        90
##
## Residual standard error: 9.859619
## Estimated effects may be unbalanced
# all pairwise comparisons among plants
# Fisher's LSD (FSD) uses "none"
pairwise.t.test(waste.long$runup, waste.long$plant,
                pool.sd = TRUE, p.adjust.method = "none")
##
##  Pairwise comparisons using t tests with pooled SD
##
## data:  waste.long$runup and waste.long$plant
##
##     PT1   PT2   PT3   PT4
## PT2 0.151 -     -     -
## PT3 0.921 0.198 -     -
## PT4 0.339 0.665 0.408 -
## PT5 0.093 0.655 0.122 0.418
##
## P value adjustment method: none
# Tukey 95% Individual p-values
TukeyHSD(fit.w)
##   Tukey multiple comparisons of means
##     95% family-wise confidence level
##
## Fit: aov(formula = runup ~ plant, data = waste.long)
##
## $plant
##               diff        lwr       upr     p adj
## PT2-PT1  4.3090909  -3.966713 12.584895 0.5976181
## PT3-PT1  0.3088517  -8.287424  8.905127 0.9999769
## PT4-PT1  2.9667464  -5.629529 11.563022 0.8718682
## PT5-PT1  5.8541958  -3.747712 15.456104 0.4408168
## PT3-PT2 -4.0002392 -12.596515  4.596036 0.6946720
## PT4-PT2 -1.3423445  -9.938620  7.253931 0.9924515
## PT5-PT2  1.5451049  -8.056803 11.147013 0.9915352
## PT4-PT3  2.6578947  -6.247327 11.563116 0.9203538
## PT5-PT3  5.5453441  -4.334112 15.424800 0.5251000
## PT5-PT4  2.8874494  -6.992007 12.766906 0.9258057
# Bonferroni 95% Individual p-values
# All Pairwise Comparisons among Levels of waste
pairwise.t.test(waste.long$runup, waste.long$plant,
                pool.sd = TRUE, p.adjust.method = "bonf")
##
##  Pairwise comparisons using t tests with pooled SD
##
## data:  waste.long$runup and waste.long$plant
```

```
##
##     PT1  PT2  PT3  PT4
## PT2 1.00 -    -    -
## PT3 1.00 1.00 -    -
## PT4 1.00 1.00 1.00 -
## PT5 0.93 1.00 1.00 1.00
##
## P value adjustment method: bonferroni
```

The residuals show many outliers

```
# QQ plot
par(mfrow=c(1,1))
library(car)
qqPlot(fit.w$residuals, las = 1, id.n = 10, id.cex = 1, lwd = 1
       , main="QQ Plot of residuals")
```

```
## 41 18 25 93 15 24 90 89 96 38
## 95 94  1  2 93  3  4 92 91 90
```



Kruskal-Wallis ANOVA is a non-parametric method for testing the hypoth-

esis of equal population medians against the alternative that not all population medians are equal. It's still not perfect here because the distributional shapes are not all the same, but it is a better alternative than the ANOVA.

```
# KW ANOVA
fit.wk <- kruskal.test(runup ~ plant, data = waste.long)
fit.wk
```

```
##
##  Kruskal-Wallis rank sum test
##
## data:  runup by plant
## Kruskal-Wallis chi-squared = 15.319, df = 4, p-value =
## 0.004084
```

```
# Bonferroni 95% pairwise comparisions with continuity correction
#   in the normal approximation for the p-value
for (i1.pt in 1:4) {
  for (i2.pt in (i1.pt+1):5) {
    wt <- wilcox.test(waste[,names(waste)[i1.pt]], waste[,names(waste)[i2.pt]]
                      , conf.int=TRUE, conf.level = 1 - 0.05/choose(5,2))
    cat(names(waste)[i1.pt], names(waste)[i2.pt])
    print(wt)
  }
}
```

```
## Warning in wilcox.test.default(waste[, names(waste)[i1.pt]], waste[, names(waste)[i2.pt]],
:  cannot compute exact p-value with ties
## Warning in wilcox.test.default(waste[, names(waste)[i1.pt]], waste[, names(waste)[i2.pt]],
:  cannot compute exact confidence intervals with ties
## PT1 PT2
##  Wilcoxon rank sum test with continuity correction
##
## data:  waste[, names(waste)[i1.pt]] and waste[, names(waste)[i2.pt]]
## W = 131.5, p-value = 0.009813
## alternative hypothesis: true location shift is not equal to 0
## 99.5 percent confidence interval:
##  -8.299958  1.599947
## sample estimates:
## difference in location
##              -4.399951
## Warning in wilcox.test.default(waste[, names(waste)[i1.pt]], waste[, names(waste)[i2.pt]],
:  cannot compute exact p-value with ties
## Warning in wilcox.test.default(waste[, names(waste)[i1.pt]], waste[, names(waste)[i2.pt]],
:  cannot compute exact confidence intervals with ties
## PT1 PT3
##  Wilcoxon rank sum test with continuity correction
##
## data:  waste[, names(waste)[i1.pt]] and waste[, names(waste)[i2.pt]]
## W = 141.5, p-value = 0.07978
```

```
## alternative hypothesis: true location shift is not equal to 0
## 99.5 percent confidence interval:
##  -6.900028  2.700029
## sample estimates:
## difference in location
##              -2.500047
## Warning in wilcox.test.default(waste[, names(waste)[i1.pt]], waste[, names(waste)[i2.pt]],
:  cannot compute exact p-value with ties
## Warning in wilcox.test.default(waste[, names(waste)[i1.pt]], waste[, names(waste)[i2.pt]],
:  cannot compute exact confidence intervals with ties
## PT1 PT4
##  Wilcoxon rank sum test with continuity correction
##
## data:  waste[, names(waste)[i1.pt]] and waste[, names(waste)[i2.pt]]
## W = 85, p-value = 0.001241
## alternative hypothesis: true location shift is not equal to 0
## 99.5 percent confidence interval:
##  -8.900056 -1.099910
## sample estimates:
## difference in location
##              -5.300051
## Warning in wilcox.test.default(waste[, names(waste)[i1.pt]], waste[, names(waste)[i2.pt]],
:  cannot compute exact p-value with ties
## Warning in wilcox.test.default(waste[, names(waste)[i1.pt]], waste[, names(waste)[i2.pt]],
:  cannot compute exact confidence intervals with ties
## PT1 PT5
##  Wilcoxon rank sum test with continuity correction
##
## data:  waste[, names(waste)[i1.pt]] and waste[, names(waste)[i2.pt]]
## W = 76, p-value = 0.02318
## alternative hypothesis: true location shift is not equal to 0
## 99.5 percent confidence interval:
##  -15.50007   3.60001
## sample estimates:
## difference in location
##              -8.703538
## Warning in wilcox.test.default(waste[, names(waste)[i1.pt]], waste[, names(waste)[i2.pt]],
:  cannot compute exact p-value with ties
## Warning in wilcox.test.default(waste[, names(waste)[i1.pt]], waste[, names(waste)[i2.pt]],
:  cannot compute exact confidence intervals with ties
## PT2 PT3
##  Wilcoxon rank sum test with continuity correction
##
## data:  waste[, names(waste)[i1.pt]] and waste[, names(waste)[i2.pt]]
## W = 238, p-value = 0.4562
## alternative hypothesis: true location shift is not equal to 0
## 99.5 percent confidence interval:
##  -3.50007  7.20000
```

```
## sample estimates:
## difference in location
##              1.352784
## Warning in wilcox.test.default(waste[, names(waste)[i1.pt]], waste[, names(waste)[i2.pt]],
:  cannot compute exact p-value with ties
## Warning in wilcox.test.default(waste[, names(waste)[i1.pt]], waste[, names(waste)[i2.pt]],
:  cannot compute exact confidence intervals with ties
## PT2 PT4
##  Wilcoxon rank sum test with continuity correction
##
## data:  waste[, names(waste)[i1.pt]] and waste[, names(waste)[i2.pt]]
## W = 186, p-value = 0.5563
## alternative hypothesis: true location shift is not equal to 0
## 99.5 percent confidence interval:
##  -5.900014  3.800023
## sample estimates:
## difference in location
##             -1.099914
## Warning in wilcox.test.default(waste[, names(waste)[i1.pt]], waste[, names(waste)[i2.pt]],
:  cannot compute exact p-value with ties
## Warning in wilcox.test.default(waste[, names(waste)[i1.pt]], waste[, names(waste)[i2.pt]],
:  cannot compute exact confidence intervals with ties
## PT2 PT5
##  Wilcoxon rank sum test with continuity correction
##
## data:  waste[, names(waste)[i1.pt]] and waste[, names(waste)[i2.pt]]
## W = 99, p-value = 0.1375
## alternative hypothesis: true location shift is not equal to 0
## 99.5 percent confidence interval:
##  -13.100001   7.400037
## sample estimates:
## difference in location
##             -4.630905
## Warning in wilcox.test.default(waste[, names(waste)[i1.pt]], waste[, names(waste)[i2.pt]],
:  cannot compute exact p-value with ties
## Warning in wilcox.test.default(waste[, names(waste)[i1.pt]], waste[, names(waste)[i2.pt]],
:  cannot compute exact confidence intervals with ties
## PT3 PT4
##  Wilcoxon rank sum test with continuity correction
##
## data:  waste[, names(waste)[i1.pt]] and waste[, names(waste)[i2.pt]]
## W = 117, p-value = 0.06583
## alternative hypothesis: true location shift is not equal to 0
## 99.5 percent confidence interval:
##  -6.800001  1.699976
## sample estimates:
## difference in location
##             -2.400038
```

```
##
## PT3 PT5
##  Wilcoxon rank sum test
##
## data:  waste[, names(waste)[i1.pt]] and waste[, names(waste)[i2.pt]]
## W = 67, p-value = 0.03018
## alternative hypothesis: true location shift is not equal to 0
## 99.5 percent confidence interval:
##  -13.4   1.7
## sample estimates:
## difference in location
##                   -6.6
## Warning in wilcox.test.default(waste[, names(waste)[i1.pt]], waste[, names(waste)[i2.pt]],
:  cannot compute exact p-value with ties
## Warning in wilcox.test.default(waste[, names(waste)[i1.pt]], waste[, names(waste)[i2.pt]],
:  cannot compute exact confidence intervals with ties
## PT4 PT5
##  Wilcoxon rank sum test with continuity correction
##
## data:  waste[, names(waste)[i1.pt]] and waste[, names(waste)[i2.pt]]
## W = 82, p-value = 0.1157
## alternative hypothesis: true location shift is not equal to 0
## 99.5 percent confidence interval:
##  -11.099965   4.799945
## sample estimates:
## difference in location
##              -4.000035
```

1.6 ADA1 Chapter 7: Categorical data analysis

Returning to the turkey dataset, below is the cross-classification of `orig` by `gt25mo`.

```
#### Example: Turkey, Chapter 7
# create a frequency table from two columns of categorical data
xt <- xtabs( ~ orig + gt25mo, data = turkey)
# display the table
xt
```

```
##     gt25mo
## orig FALSE TRUE
##   va     2    6
##   wi     3    4
```

```
# summary from xtabs() is the same as chisq.test() without continuity correction
```

```
summary(xt)
```

```
## Call: xtabs(formula = ~orig + gt25mo, data = turkey)
## Number of cases in table: 15
## Number of factors: 2
## Test for independence of all factors:
##  Chisq = 0.5357, df = 1, p-value = 0.4642
##  Chi-squared approximation may be incorrect
```

```
# same as xtabs()
x.summary <- chisq.test(xt, correct=FALSE)
```

```
## Warning in chisq.test(xt, correct = FALSE): Chi-squared approximation may be incorrect
```

```
x.summary
```

```
##
##  Pearson's Chi-squared test
##
## data:  xt
## X-squared = 0.53571, df = 1, p-value = 0.4642
```

```
# the default is to perform Yates' continuity correction
chisq.test(xt)
```

```
## Warning in chisq.test(xt):  Chi-squared approximation may be incorrect
```

```
##
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  xt
## X-squared = 0.033482, df = 1, p-value = 0.8548
```

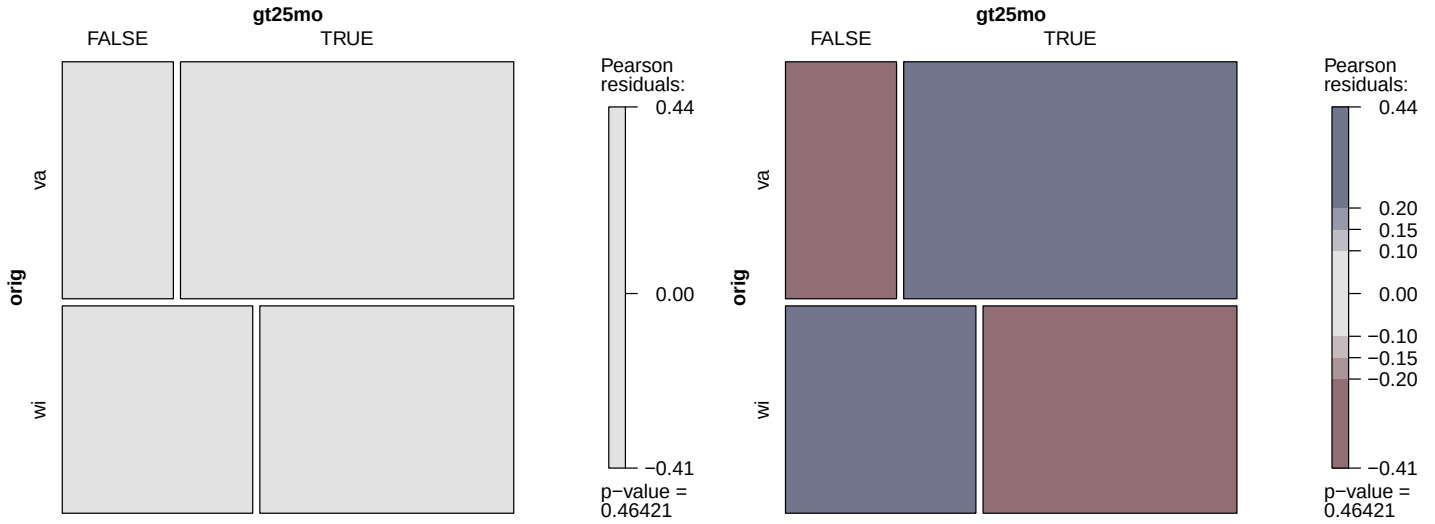
```
# Fisher's exact test
fisher.test(xt)
```

```
##
##  Fisher's Exact Test for Count Data
##
## data:  xt
## p-value = 0.6084
## alternative hypothesis: true odds ratio is not equal to 1
## 95 percent confidence interval:
##  0.02687938 6.23767632
## sample estimates:
## odds ratio
##  0.4698172
```

A mosaic plot is for categorical data. Area represents frequency. The default shading is a good start, since colors only appear when there's evidence of association related to those cell values. In our example, there's insufficient evidence for association, so the default shading is all gray.

```
library(vcd)    # for mosaic()
# shading based on significance relative to appropriate chi-square distribution
```

```
mosaic(xt, shade=TRUE)
# you can define your own interpolated shading
mosaic(xt, shade=TRUE, gp_args = list(interpolate = seq(.1,.2,.05)))
```



From the `chisq.test()` above, we make a table to summarize important values from that analysis and compare the observed and expected frequencies in plots.

```
# use output in x.summary and create table
x.table <- data.frame(obs = x.summary$observed
                      , exp = x.summary$expected
                      , res = x.summary$residuals
                      , chisq = x.summary$residuals^2
                      , stdres = x.summary$stdres)
```

```
## Warning in data.frame(obs = x.summary$observed, exp = x.summary$expected, :  row names
were found from a short variable and have been discarded
```

```
# There are duplicate row and col identifiers in this x.table
#   because we're creating vectors from a two-way table
#   and columns identifying row and col names are automatically added.
# Can you figure out the naming scheme?
x.table
```

```
##   obs.orig obs.gt25mo obs.Freq exp.FALSE exp.TRUE res.orig res.gt25mo
## 1       va      FALSE        2  2.666667 5.333333       va      FALSE
## 2       wi      FALSE        3  2.333333 4.666667       wi      FALSE
## 3       va       TRUE        6  2.666667 5.333333       va       TRUE
## 4       wi       TRUE        4  2.333333 4.666667       wi       TRUE
##     res.Freq chisq.orig chisq.gt25mo chisq.Freq stdres.orig
## 1 -0.4082483         va        FALSE 0.16666667          va
```

```
## 2  0.4364358         wi        FALSE 0.19047619          wi
## 3  0.2886751         va         TRUE 0.08333333          va
## 4 -0.3086067         wi         TRUE 0.09523810          wi
##   stdres.gt25mo stdres.Freq
## 1         FALSE  -0.7319251
## 2         FALSE   0.7319251
## 3          TRUE   0.7319251
## 4          TRUE  -0.7319251
```

```
# create a single column with a joint cell name
x.table$cellname <- paste(as.character(x.table$obs.orig)
                        , as.character(x.table$obs.gt25mo)
                        , sep="_")
# expected frequencies in a single column
x.table$exp <- c(x.table$exp.FALSE[1:2], x.table$exp.TRUE[3:4])
# create a simpler name for the obs, res, chisq, stdres
x.table$obs    <- x.table$obs.Freq
x.table$res    <- x.table$res.Freq
x.table$chisq  <- x.table$chisq.Freq
x.table$stdres <- x.table$stdres.Freq

# include only the "cleaned" columns
x.table <- subset(x.table, select = c(cellname, obs, exp, res, chisq, stdres))
x.table
```

```
##   cellname obs      exp        res      chisq     stdres
## 1 va_FALSE   2 2.666667 -0.4082483 0.16666667 -0.7319251
## 2 wi_FALSE   3 2.333333  0.4364358 0.19047619  0.7319251
## 3  va_TRUE   6 5.333333  0.2886751 0.08333333  0.7319251
## 4  wi_TRUE   4 4.666667 -0.3086067 0.09523810 -0.7319251
```

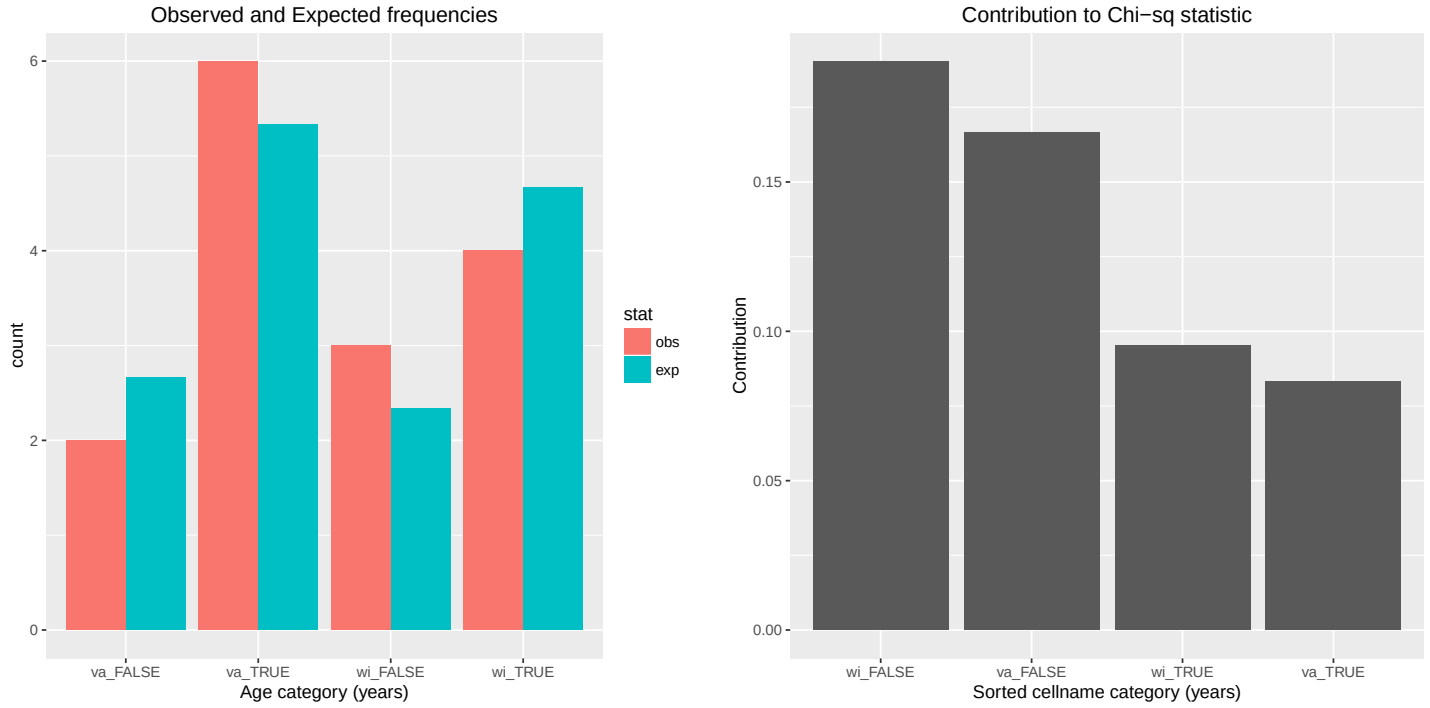
```
# reshape the data for plotting
library(reshape2)
x.table.obsexp <- melt(x.table,
              # id.vars: ID variables
              #   all variables to keep but not split apart on
              id.vars=c("cellname"),
              # measure.vars: The source columns
              #   (if unspecified then all other variables are measure.vars)
              measure.vars = c("obs", "exp"),
              # variable.name: Name of the destination column identifying each
              #   original column that the measurement came from
              variable.name = "stat",
              # value.name: column name for values in table
              value.name = "value"
            )
x.table.obsexp
```

```
##   cellname stat    value
## 1 va_FALSE  obs 2.000000
## 2 wi_FALSE  obs 3.000000
```

```
## 3  va_TRUE  obs 6.000000
## 4  wi_TRUE  obs 4.000000
## 5 va_FALSE  exp 2.666667
## 6 wi_FALSE  exp 2.333333
## 7  va_TRUE  exp 5.333333
## 8  wi_TRUE  exp 4.666667
```

Plot observed vs expected frequencies, and the contribution to chi-square statistic sorted decending.

```
# Observed vs Expected counts
library(ggplot2)
p <- ggplot(x.table.obsexp, aes(x = cellname, fill = stat, weight=value))
p <- p + geom_bar(position="dodge")
p <- p + labs(title = "Observed and Expected frequencies")
p <- p + xlab("Age category (years)")
print(p)


# Contribution to chi-sq
# pull out only the cellname and chisq columns
x.table.chisq <- x.table[, c("cellname","chisq")]
# reorder the cellname categories to be descending relative to the chisq statistic
x.table.chisq$cellname <- with(x.table, reorder(cellname, -chisq))

p <- ggplot(x.table.chisq, aes(x = cellname, weight = chisq))
p <- p + geom_bar()
p <- p + labs(title = "Contribution to Chi-sq statistic")
p <- p + xlab("Sorted cellname category (years)")
p <- p + ylab("Contribution")
print(p)
```



1.7 ADA1 Chapter 8: Correlation and regression

Rocket Propellant Data A rocket motor is manufactured by bonding an igniter propellant and a sustainer propellant together inside a metal housing. The shear strength of the bond between the two types of propellant is an important quality characteristic. It is suspected that shear strength is related to the age in weeks of the batch of sustainer propellant. Twenty observations on these two characteristics are given below. The first column is shear strength in psi, the second is age of propellant in weeks.

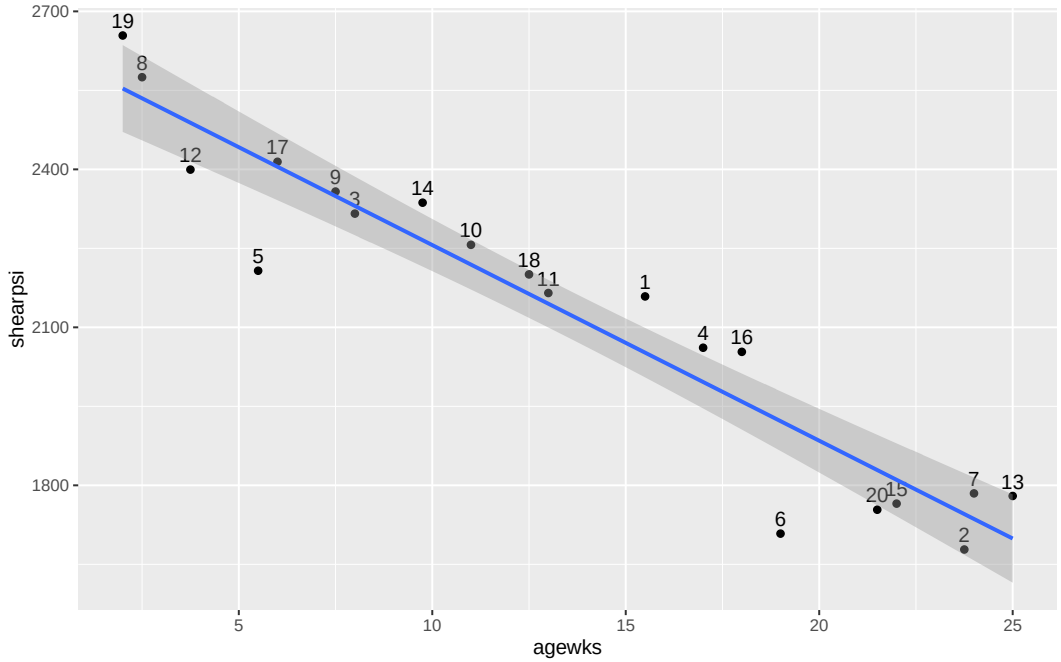
```
#### Example: Rocket, Chapter 8
fn.data <- "http://statacumen.com/teach/ADA2/ADA2_notes_Ch01_rocket.dat"
# this file uses spaces as delimiters, so use read.table()
rocket <- read.table(fn.data, header = TRUE)
rocket$id <- 1:nrow(rocket) # add an id variable to identify observations
str(rocket)
```

```
## 'data.frame': 20 obs. of  3 variables:
##  $ shearpsi: num  2159 1678 2316 2061 2208 ...
##  $ agewks  : num  15.5 23.8 8 17 5.5 ...
##  $ id      : int  1 2 3 4 5 6 7 8 9 10 ...
```

```
head(rocket)
```

```
##   shearpsi agewks id
## 1  2158.70  15.50  1
## 2  1678.15  23.75  2
## 3  2316.00   8.00  3
## 4  2061.30  17.00  4
## 5  2207.50   5.50  5
## 6  1708.30  19.00  6
```

```
# ggplot: Plot the data with linear regression fit and confidence bands
library(ggplot2)
p <- ggplot(rocket, aes(x = agewks, y = shearpsi, label = id))
p <- p + geom_point()
# plot labels next to points
p <- p + geom_text(hjust = 0.5, vjust = -0.5)
# plot regression line and confidence band
p <- p + geom_smooth(method = lm)
print(p)
```



The data are reasonably linear, so fit the regression.

```
# fit the simple linear regression model
lm.shearpsi.agewks <- lm(shearpsi ~ agewks, data = rocket)
# use summary() to get t-tests of parameters (slope, intercept)
summary(lm.shearpsi.agewks)
```

```
##
## Call:
## lm(formula = shearpsi ~ agewks, data = rocket)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -215.98  -50.68   28.74   66.61  106.76
##
```

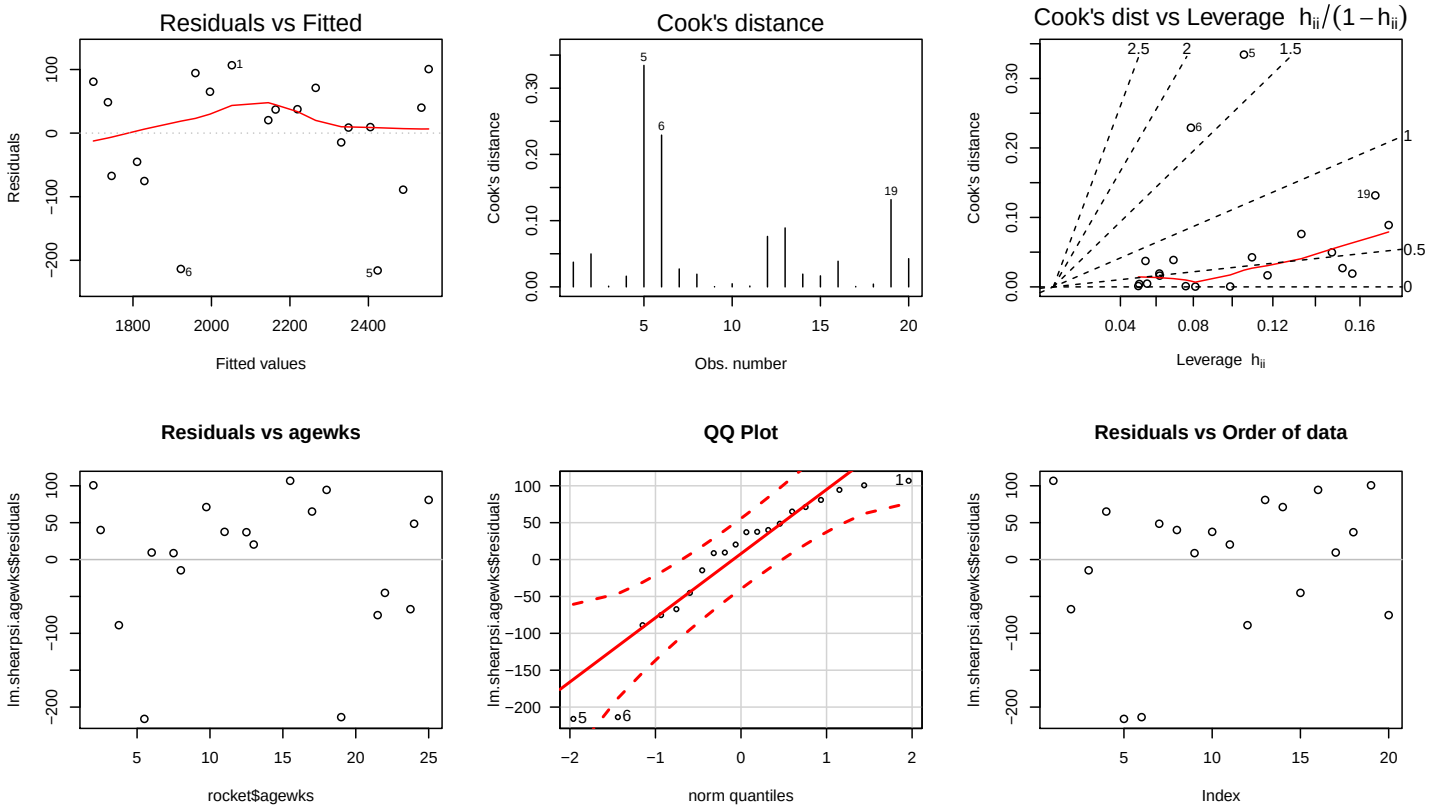
```
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 2627.822     44.184   59.48  < 2e-16 ***
## agewks       -37.154      2.889  -12.86 1.64e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 96.11 on 18 degrees of freedom
## Multiple R-squared:  0.9018,Adjusted R-squared:  0.8964
## F-statistic: 165.4 on 1 and 18 DF,  p-value: 1.643e-10
```

Plot diagnostics.

```
# plot diagnistics
par(mfrow=c(2,3))
plot(lm.shearpsi.agewks, which = c(1,4,6))

# residuals vs weight
plot(rocket$agewks, lm.shearpsi.agewks$residuals, main="Residuals vs agewks")
  # horizontal line at zero
  abline(h = 0, col = "gray75")

# Normality of Residuals
library(car)
qqPlot(lm.shearpsi.agewks$residuals, las = 1, id.n = 3, main="QQ Plot")
```

```
##  5  6  1
##  1  2 20
```

```
# residuals vs order of data
plot(lm.shearpsi.agewks$residuals, main="Residuals vs Order of data")
  # horizontal line at zero
  abline(h = 0, col = "gray75")
```



The relationship between shear strength and age is fairly linear with predicted shear strength decreasing as the age of the propellant increases. The fitted LS line is

$$\text{Predicted shear strength} = 2627.8 - 37.2\ \text{Age}.$$

The test for $H_0 : \beta_1 = 0$ (zero slope for the population regression line) is highly significant: p-value< 0.0001. Also note that $R^2 = 0.9018$ so the linear relationship between shear strength and age explains about 90% of the variation in shear strength.

The data plot and residual information identify observations 5 and 6 as potential outliers ($r_5 = -2.38, r_6 = -2.32$). The predicted values for these observations are much greater than the observed shear strengths. These same observations appear as potential outliers in the normal scores plot and the plot of r_i against $\hat{Y}_i$. Observations 5 and 6 also have the largest influence on the analysis; see the Cook's distance values.

A sensible next step would be to repeat the analysis holding out the most influential case, observation 5. It should be somewhat clear that the influence

of case 6 would increase dramatically once case 5 is omitted from the analysis. Since both cases have essentially the same effect on the positioning of the LS line, I will assess the impact of omitting both simultaneously.

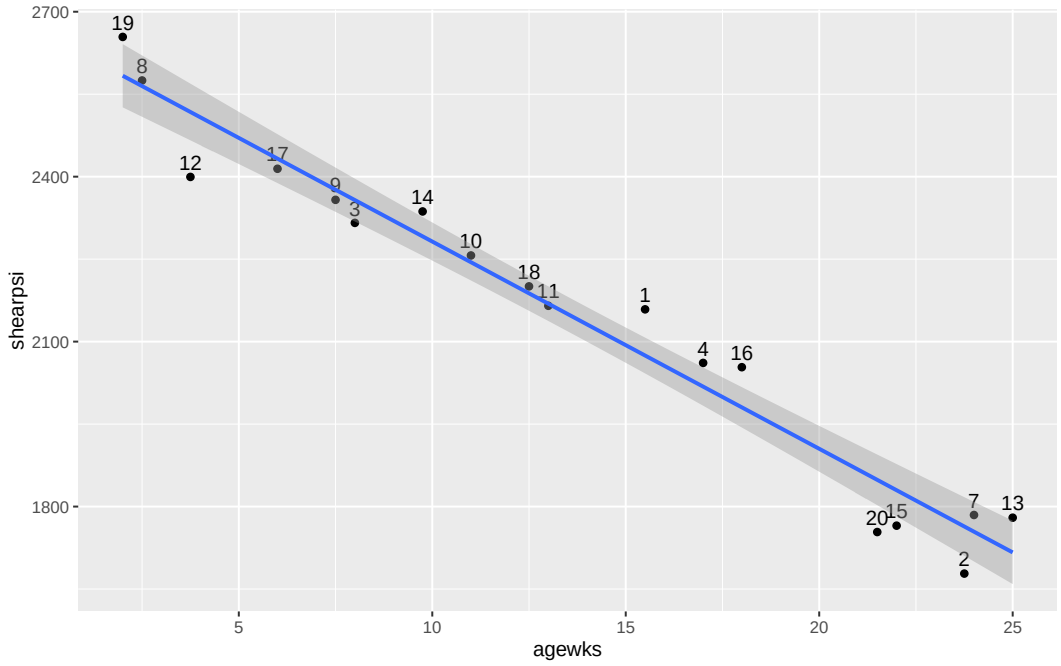
Before we hold out these cases, how do you think the LS line will change? My guess is these cases are pulling the LS line down, so the intercept of the LS line should increase once these cases are omitted. Holding out either case 5 or 6 would probably also affect the slope, but my guess is that when they are both omitted the slope will change little. (Is this my experience speaking, or have I already seen the output? Both.) What will happen to R^2 when we delete these points?

Exclude observations 5 and 6 and redo the analysis.

```
# exclude observations 5 and 6
rocket56 <- rocket[-c(5,6), ]
head(rocket56)
```

```
##   shearpsi agewks id
## 1  2158.70  15.50  1
## 2  1678.15  23.75  2
## 3  2316.00   8.00  3
## 4  2061.30  17.00  4
## 7  1784.70  24.00  7
## 8  2575.00   2.50  8
```

```
# ggplot: Plot the data with linear regression fit and confidence bands
library(ggplot2)
p <- ggplot(rocket56, aes(x = agewks, y = shearpsi, label = id))
p <- p + geom_point()
# plot labels next to points
p <- p + geom_text(hjust = 0.5, vjust = -0.5)
# plot regression line and confidence band
p <- p + geom_smooth(method = lm)
print(p)
```



The data are reasonably linear, so fit the regression.

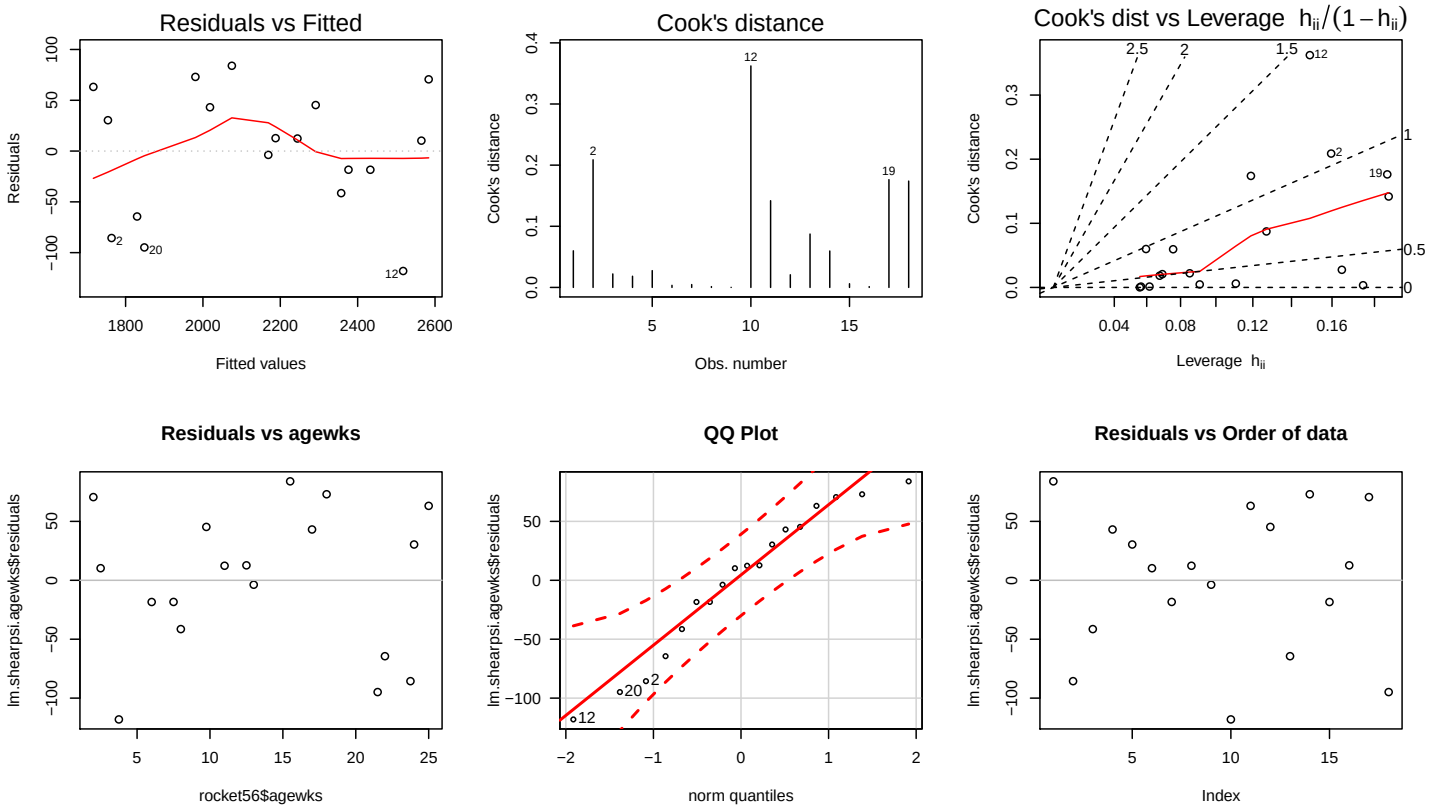
```
# fit the simple linear regression model
lm.shearpsi.agewks <- lm(shearpsi ~ agewks, data = rocket56)
# use summary() to get t-tests of parameters (slope, intercept)
summary(lm.shearpsi.agewks)
```

```
##
## Call:
## lm(formula = shearpsi ~ agewks, data = rocket56)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -118.07  -35.67   11.31   44.75   83.98
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 2658.973     30.533   87.08  < 2e-16 ***
## agewks       -37.694      1.979  -19.05 2.02e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 62.97 on 16 degrees of freedom
## Multiple R-squared:  0.9578,Adjusted R-squared:  0.9551
## F-statistic: 362.9 on 1 and 16 DF,  p-value: 2.023e-12
```

Plot diagnostics.

```
# plot diagnistics
par(mfrow=c(2,3))
plot(lm.shearpsi.agewks, which = c(1,4,6))

# residuals vs weight
plot(rocket56$agewks, lm.shearpsi.agewks$residuals, main="Residuals vs agewks")
```

```
  # horizontal line at zero
  abline(h = 0, col = "gray75")

# Normality of Residuals
library(car)
qqPlot(lm.shearpsi.agewks$residuals, las = 1, id.n = 3, main="QQ Plot")
## 12 20  2
##  1  2  3
# residuals vs order of data
plot(lm.shearpsi.agewks$residuals, main="Residuals vs Order of data")
  # horizontal line at zero
  abline(h = 0, col = "gray75")
```



Some summaries for the complete analysis, and when cases 5 and 6 are held out, are given below. The summaries lead to the following conclusions:

1. Holding out cases 5 and 6 has little effect on the estimated LS line. Predictions of shear strength are slightly larger after holding out these two cases (recall that intercept increased, but slope was roughly the same!)
2. Holding out these two cases decreases $\hat{\sigma}$ considerably, and leads to a modest increase in R^2. The complete data set will give wider CI and prediction intervals than the analysis which deletes case 5 and 6 because

$\hat{\sigma}$ decreases when these points are omitted.

3. Once these cases are held out, the normal scores plot and plot of the studentized residuals against fitted values shows no significant problems. One observation has a large Cook's D but does not appear to be extremely influential.

Without any substantive reason to explain the low shear strengths for cases 5 and 6, I am hesitant to delete either case from the analysis. I feel relatively confident that including these cases will not seriously limit the use of the model.

Feature	Full data	Omit 5 and 6
b_0	2627.82	2658.97
b_1	−37.15	−37.69
R^2	0.9018	0.9578
$\hat{\sigma}$	96.10	62.96
p-val for $H_0 : \beta_1 = 0$	0.0001	0.0001

Here is a comparison of the predicted or fitted values for selected observations in the data set, based on the two fits.

Observation	Actual Shear Strength	Pred, full	Pred, omit 5,6
1	2159	2052	2075
2	1678	1745	1764
4	2061	1996	2018
8	2575	2535	2565
10	2257	2219	2244
15	1765	1810	1830
18	2201	2163	2188
20	1754	1829	1849

Review complete

Now that we're warmed up, let's dive into new material!