SIOB 296 Introduction to Programming with R

Eric Archer (eric.archer@noaa.gov)

Week 09: March 2, 2020

Distributions

Functions are provided to calculate the density, distribution function, quantile function, and generate random numbers from a variety of parametric distributions. They have similar forms, where if `<stat>` is the name of the distribution (e.g., `norm` for Normal, `unif` for Uniform, `binom` for Binomial), `d<stat>` gives the density or probability mass function (likelihood), `p<stat>` gives the probability distribution (cumulative distribution function), `q<stat>` gives the quantile function, and `r<stat>` generates random numbers. Here are examples of all for for the Normal distribution:

```
# The likelihood of five values in a Normal distribution with a
#  mean of 10 and a standard deviation of 2:
x <- c(5, 8, 10, 12, 15)
dnorm(x, mean = 10, sd = 2)
```

```
[1] 0.00876415 0.12098536 0.19947114 0.12098536 0.00876415
```

```
# Cumulative probability of same values:
pnorm(x, mean = 10, sd = 2)
```

```
[1] 0.006209665 0.158655254 0.500000000 0.841344746 0.993790335
```

```
# Quantiles:
p <- c(0.05, 0.25, 0.5, 0.75, 0.95)
qnorm(p, mean = 10, sd = 2)
```

```
[1]  6.710293  8.651020 10.000000 11.348980 13.289707
```

```
# Five random draws:
rnorm(5, mean = 10, sd = 2)
```

```
[1] 11.37633 10.64688 11.89854 11.98003 12.98298
```

The random number seed is set with `set.seed()`. Setting this value ensures that the same random number sequence will be repeated:

```
# repeat the same random 5 numbers
set.seed(1)
rnorm(5, mean = 10, sd = 2)
```

```
[1]  8.747092 10.367287  8.328743 13.190562 10.659016
```

```
set.seed(1)
rnorm(5, mean = 10, sd = 2)
```

```
[1]  8.747092 10.367287  8.328743 13.190562 10.659016
```

```
# choose a different random 5
rnorm(5, mean = 10, sd = 2)
```

```
[1]  8.359063 10.974858 11.476649 11.151563  9.389223
```

Statistical tests

There are several functions for standard statistical tests that all have similar outputs. The most common ones are `binom.test`, `chisq.test`, `kruskal.test`, `ks.test`, and `t.test`. As an example, we'll simulate two sets of length measurements and conduct a t-test to test for differences between their means.

```
# choose a number of individuals from each species
n.ind <- sample(30:300, 1)

# simulate drawing some lengths from a normal distribution
spp1 <- rnorm(n.ind, 10, 3)
spp2 <- rnorm(n.ind, 11, 3)

# test difference between means
spp.ttest <- t.test(spp1, spp2)
spp.ttest
```

```

    Welch Two Sample t-test

data:  spp1 and spp2
t = -1.5726, df = 210.03, p-value = 0.1173
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -1.3877880  0.1561224
sample estimates:
mean of x mean of y
 10.29191  10.90774
```

```
str(spp.ttest)
```

```
List of 10
 $ statistic  : Named num -1.57
  ..- attr(*, "names")= chr "t"
 $ parameter  : Named num 210
  ..- attr(*, "names")= chr "df"
 $ p.value    : num 0.117
 $ conf.int   : num [1:2] -1.388 0.156
  ..- attr(*, "conf.level")= num 0.95
 $ estimate   : Named num [1:2] 10.3 10.9
  ..- attr(*, "names")= chr [1:2] "mean of x" "mean of y"
 $ null.value : Named num 0
  ..- attr(*, "names")= chr "difference in means"
 $ stderr     : num 0.392
 $ alternative: chr "two.sided"
 $ method     : chr "Welch Two Sample t-test"
 $ data.name  : chr "spp1 and spp2"
 - attr(*, "class")= chr "htest"
```

Note that the result of the `t.test` function is a list with various results and information about the test conducted. Because the function returns a named list, if we are only interested in one element, we can extract it directly:

```
t.test(spp1, spp2)$p.value
```

```
[1] 0.1173081
```

Another example of a common test is a chi-squared test of differences among frequencies, run with the function `chisq.test`. Lets also simulate some data of iris species occurence using the `sample` function. We'll give different weights to the `prob` argument to vary the occurence of species in each plot.

```
# collect samples from plot 1
plot1.n <- sample(30:100, 1)
plot1 <- sample(levels(iris$Species), plot1.n, T, c(1, 2, 5))

# collect samples from plot 2
plot2.n <- sample(30:100, 1)
plot2 <- sample(levels(iris$Species), plot2.n, T, c(2, 1, 5))

# create a data.frame combining plot samples
occ.df <- rbind(
  cbind(spp = plot1, plot = 1),
  cbind(spp = plot2, plot = 2)
)

table(occ.df[, "spp"], occ.df[, "plot"])
```

```
             1  2
  setosa     5 15
  versicolor  7  4
  virginica  20 28
```

```
occ.chisq <- chisq.test(occ.df[, "spp"], occ.df[, "plot"])
```

```
Warning in chisq.test(occ.df[, "spp"], occ.df[, "plot"]): Chi-squared
approximation may be incorrect
```

```
occ.chisq
```

```
    Pearson's Chi-squared test

data:  occ.df[, "spp"] and occ.df[, "plot"]
X-squared = 4.4644, df = 2, p-value = 0.1073
```

```
str(occ.chisq)
```

```
List of 9
 $ statistic: Named num 4.46
  ..- attr(*, "names")= chr "X-squared"
 $ parameter: Named int 2
  ..- attr(*, "names")= chr "df"
 $ p.value  : num 0.107
 $ method   : chr "Pearson's Chi-squared test"
 $ data.name: chr "occ.df[, \"spp\"] and occ.df[, \"plot\"]"
 $ observed : 'table' int [1:3, 1:2] 5 7 20 15 4 28
  ..- attr(*, "dimnames")=List of 2
  .. ..$ occ.df[, "spp"] : chr [1:3] "setosa" "versicolor" "virginica"
  .. ..$ occ.df[, "plot"]: chr [1:2] "1" "2"
 $ expected : num [1:3, 1:2] 8.1 4.46 19.44 11.9 6.54 ...
  ..- attr(*, "dimnames")=List of 2
```

```
  .. ..$ occ.df[, "spp"] : chr [1:3] "setosa" "versicolor" "virginica"
  .. ..$ occ.df[, "plot"]: chr [1:2] "1" "2"
 $ residuals: 'table' num [1:3, 1:2] -1.09 1.205 0.126 0.899 -0.995 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ occ.df[, "spp"] : chr [1:3] "setosa" "versicolor" "virginica"
  .. ..$ occ.df[, "plot"]: chr [1:2] "1" "2"
 $ stdres   : 'table' num [1:3, 1:2] -1.635 1.684 0.261 1.635 -1.684 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ occ.df[, "spp"] : chr [1:3] "setosa" "versicolor" "virginica"
  .. ..$ occ.df[, "plot"]: chr [1:2] "1" "2"
 - attr(*, "class")= chr "htest"
```

The `chisq.test` function also has the ability to estimate significance via a bootstrap, which is selected by setting `simulate.p.value = TRUE`:

```
chisq.test(occ.df[, "spp"], occ.df[, "plot"], sim = TRUE)
```

```
    Pearson's Chi-squared test with simulated p-value (based on 2000
    replicates)

data:  occ.df[, "spp"] and occ.df[, "plot"]
X-squared = 4.4644, df = NA, p-value = 0.1259
```

We can directly calculate the Chi-squared statistic using some matrix algebra and a few summary functions. Recall that Chi-squared = sum((observed - expected)^2 / expected)

```
# observed frequencies
obs <- table(occ.df[, "spp"], occ.df[, "plot"])
obs
```

```
             1  2
  setosa     5 15
  versicolor  7  4
  virginica  20 28
```

```
# row sums and column sums
row.sums <- rowSums(obs)
row.sums
```

```
    setosa versicolor  virginica
        20         11         48
```

```
col.sums <- colSums(obs)
col.sums
```

```
 1  2
32 47
```

```
# expected frequencies are the matrix product of these two divided by the total
exp <- outer(row.sums, col.sums) / sum(freq)
```

```
Error in eval(expr, envir, enclos): object 'freq' not found
```

```
chi.squared <- sum((obs - exp) ^ 2 / exp)
```

```
Error in obs - exp: non-numeric argument to binary operator
```

```
chi.squared
```

```
Error in eval(expr, envir, enclos): object 'chi.squared' not found
```

```
# compared to value from chisq.test function:
occ.chisq$statistic
```

```
X-squared
 4.464363
```

```
# the p-value of this can be looked up from the chisq.distribution
1 - pchisq(chi.squared, df = 2)
```

```
Error in pchisq(chi.squared, df = 2): object 'chi.squared' not found
```

Formula

In R, models are usually based on `formula` objects. Formulae are constructed using the tilde (`~`) operator. The syntax is `y ~ x`, which is translated as `y` is a function of `x`. As an example, we can create a data frame of our simulated length data and run the t-test using a formula structure instead.

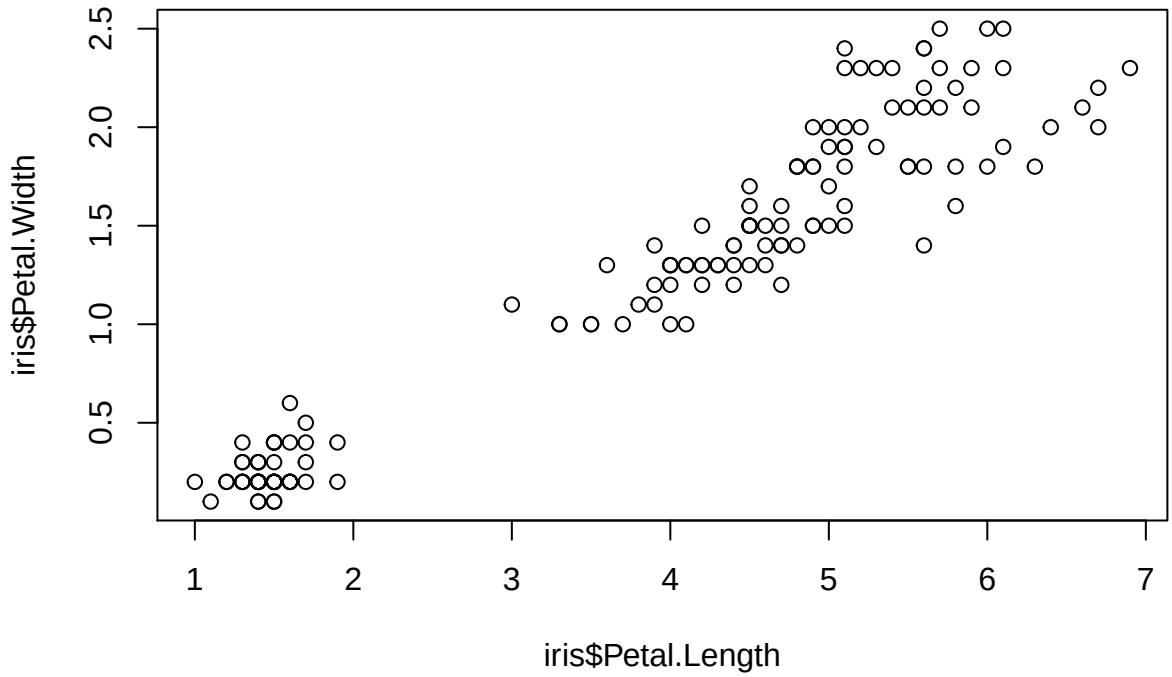
```
# create data frame
length.df <- data.frame(
  length = c(spp1, spp2),
  spp = rep(c(1, 2), each = n.ind)
)

# run t-test with formula based on columns in data frame
t.test(length ~ spp, data = length.df)
```

```
    Welch Two Sample t-test

data:  length by spp
t = -1.5726, df = 210.03, p-value = 0.1173
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -1.3877880  0.1561224
sample estimates:
mean in group 1 mean in group 2
       10.29191        10.90774
```

Linear models

We also use formula for linear and non-linear modelling. The standard function `lm` fits a linear model and returns the intercept and slope estimates as well as diagnostics of the fit. Below, we'll fit a model to estimate iris petal width from length.

```
plot(iris$Petal.Length, iris$Petal.Width)
```




```r
# Fit the model
petal.lm <- lm(Petal.Width ~ Petal.Length, data = iris)
# Here's a simple summary of the fit
petal.lm
```

```
Call:
lm(formula = Petal.Width ~ Petal.Length, data = iris)

Coefficients:
 (Intercept)  Petal.Length
     -0.3631        0.4158
```

```r
# Here are all of the elements in the fitted object:
str(petal.lm)
```

```
List of 12
 $ coefficients : Named num [1:2] -0.363 0.416
  ..- attr(*, "names")= chr [1:2] "(Intercept)" "Petal.Length"
 $ residuals    : Named num [1:150] -0.019 -0.019 0.0226 -0.0606 -0.019 ...
  ..- attr(*, "names")= chr [1:150] "1" "2" "3" "4" ...
 $ effects      : Named num [1:150] -14.6888 8.9588 0.0257 -0.0576 -0.0159 ...
  ..- attr(*, "names")= chr [1:150] "(Intercept)" "Petal.Length" "" "" ...
 $ rank         : int 2
 $ fitted.values: Named num [1:150] 0.219 0.219 0.177 0.261 0.219 ...
  ..- attr(*, "names")= chr [1:150] "1" "2" "3" "4" ...
 $ assign       : int [1:2] 0 1
 $ qr           :List of 5
  ..$ qr   : num [1:150, 1:2] -12.2474 0.0816 0.0816 0.0816 0.0816 ...
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : chr [1:150] "1" "2" "3" "4" ...
  .. .. ..$ : chr [1:2] "(Intercept)" "Petal.Length"
  .. ..- attr(*, "assign")= int [1:2] 0 1
```

```
  ..$ qraux: num [1:2] 1.08 1.1
  ..$ pivot: int [1:2] 1 2
  ..$ tol  : num 1e-07
  ..$ rank : int 2
  ..- attr(*, "class")= chr "qr"
 $ df.residual  : int 148
 $ xlevels      : Named list()
 $ call         : language lm(formula = Petal.Width ~ Petal.Length, data = iris)
 $ terms        :Classes 'terms', 'formula'  language Petal.Width ~ Petal.Length
  .. ..- attr(*, "variables")= language list(Petal.Width, Petal.Length)
  .. ..- attr(*, "factors")= int [1:2, 1] 0 1
  .. .. ..- attr(*, "dimnames")=List of 2
  .. .. .. ..$ : chr [1:2] "Petal.Width" "Petal.Length"
  .. .. .. ..$ : chr "Petal.Length"
  .. ..- attr(*, "term.labels")= chr "Petal.Length"
  .. ..- attr(*, "order")= int 1
  .. ..- attr(*, "intercept")= int 1
  .. ..- attr(*, "response")= int 1
  .. ..- attr(*, ".Environment")=<environment: R_GlobalEnv>
  .. ..- attr(*, "predvars")= language list(Petal.Width, Petal.Length)
  .. ..- attr(*, "dataClasses")= Named chr [1:2] "numeric" "numeric"
  .. .. ..- attr(*, "names")= chr [1:2] "Petal.Width" "Petal.Length"
 $ model        :'data.frame':  150 obs. of  2 variables:
  ..$ Petal.Width : num [1:150] 0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
  ..$ Petal.Length: num [1:150] 1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
  ..- attr(*, "terms")=Classes 'terms', 'formula'  language Petal.Width ~ Petal.Length
  .. .. ..- attr(*, "variables")= language list(Petal.Width, Petal.Length)
  .. .. ..- attr(*, "factors")= int [1:2, 1] 0 1
  .. .. .. ..- attr(*, "dimnames")=List of 2
  .. .. .. .. ..$ : chr [1:2] "Petal.Width" "Petal.Length"
  .. .. .. .. ..$ : chr "Petal.Length"
  .. .. ..- attr(*, "term.labels")= chr "Petal.Length"
  .. .. ..- attr(*, "order")= int 1
  .. .. ..- attr(*, "intercept")= int 1
  .. .. ..- attr(*, "response")= int 1
  .. .. ..- attr(*, ".Environment")=<environment: R_GlobalEnv>
  .. .. ..- attr(*, "predvars")= language list(Petal.Width, Petal.Length)
  .. .. ..- attr(*, "dataClasses")= Named chr [1:2] "numeric" "numeric"
  .. .. .. ..- attr(*, "names")= chr [1:2] "Petal.Width" "Petal.Length"
 - attr(*, "class")= chr "lm"
```

Elements can be extracted from this list by name. For example, the estimated coefficients are stored in the `$coefficients` element:

```
petal.lm$coefficients
```

```
 (Intercept) Petal.Length
  -0.3630755    0.4157554
```

However, there are a set of functions for extracting common elements from model fits. An example is the `coef()` function, which will also extract the coefficients:

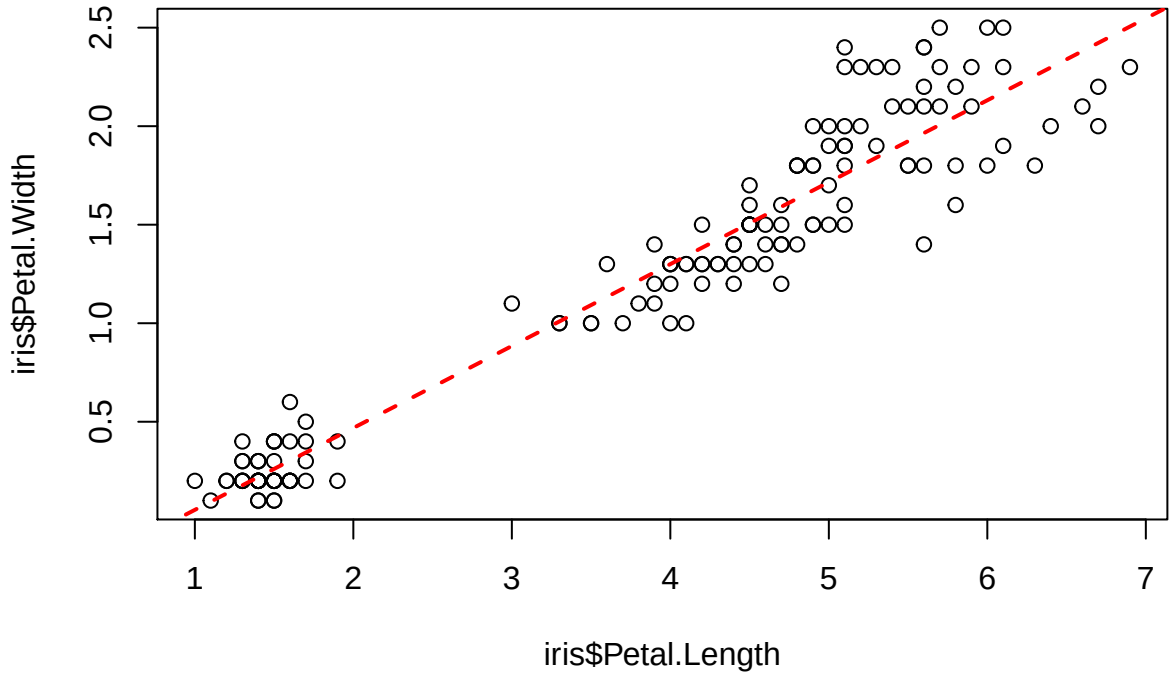
```
coef(petal.lm)
```

```
 (Intercept) Petal.Length
  -0.3630755    0.4157554
```

Others are `residuals` and `fitted.values`, which will often work with model objects from other routines such as `glm` and `nls`.

We can use the `abline` function to plot the estimated fit over the data:

```
plot(iris$Petal.Length, iris$Petal.Width)
abline(petal.lm, col = "red", lwd = 2, lty = "dashed")
```



More detail about the fit can be extracted with the `summary` function. In particular, we can see a summary of the residuals to inspect normality of the errors, as well as the standard errors and p-values for tests of significant deviation of the estimated parameters from zero:

```
lm.smry <- summary(petal.lm)
print(lm.smry)
```

```
Call:
lm(formula = Petal.Width ~ Petal.Length, data = iris)

Residuals:
     Min       1Q   Median       3Q      Max
-0.56515 -0.12358 -0.01898  0.13288  0.64272

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  -0.363076   0.039762  -9.131  4.7e-16 ***
Petal.Length  0.415755   0.009582  43.387  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.2065 on 148 degrees of freedom
Multiple R-squared:  0.9271,    Adjusted R-squared:  0.9266
F-statistic:  1882 on 1 and 148 DF,  p-value: < 2.2e-16
```

```
str(lm.smry)
```

```
List of 11
 $ call         : language lm(formula = Petal.Width ~ Petal.Length, data = iris)
 $ terms        :Classes 'terms', 'formula'  language Petal.Width ~ Petal.Length
  .. ..- attr(*, "variables")= language list(Petal.Width, Petal.Length)
  .. ..- attr(*, "factors")= int [1:2, 1] 0 1
  .. .. ..- attr(*, "dimnames")=List of 2
  .. .. .. ..$ : chr [1:2] "Petal.Width" "Petal.Length"
  .. .. .. ..$ : chr "Petal.Length"
  .. ..- attr(*, "term.labels")= chr "Petal.Length"
  .. ..- attr(*, "order")= int 1
  .. ..- attr(*, "intercept")= int 1
  .. ..- attr(*, "response")= int 1
  .. ..- attr(*, ".Environment")=<environment: R_GlobalEnv>
  .. ..- attr(*, "predvars")= language list(Petal.Width, Petal.Length)
  .. ..- attr(*, "dataClasses")= Named chr [1:2] "numeric" "numeric"
  .. .. ..- attr(*, "names")= chr [1:2] "Petal.Width" "Petal.Length"
 $ residuals    : Named num [1:150] -0.019 -0.019 0.0226 -0.0606 -0.019 ...
  ..- attr(*, "names")= chr [1:150] "1" "2" "3" "4" ...
 $ coefficients : num [1:2, 1:4] -0.36308 0.41576 0.03976 0.00958 -9.13122 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:2] "(Intercept)" "Petal.Length"
  .. ..$ : chr [1:4] "Estimate" "Std. Error" "t value" "Pr(>|t|)"
 $ aliased      : Named logi [1:2] FALSE FALSE
  ..- attr(*, "names")= chr [1:2] "(Intercept)" "Petal.Length"
 $ sigma        : num 0.206
 $ df           : int [1:3] 2 148 2
 $ r.squared    : num 0.927
 $ adj.r.squared: num 0.927
 $ fstatistic   : Named num [1:3] 1882 1 148
  ..- attr(*, "names")= chr [1:3] "value" "numdf" "dendf"
 $ cov.unscaled : num [1:2, 1:2] 0.03708 -0.00809 -0.00809 0.00215
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:2] "(Intercept)" "Petal.Length"
  .. ..$ : chr [1:2] "(Intercept)" "Petal.Length"
 - attr(*, "class")= chr "summary.lm"
```

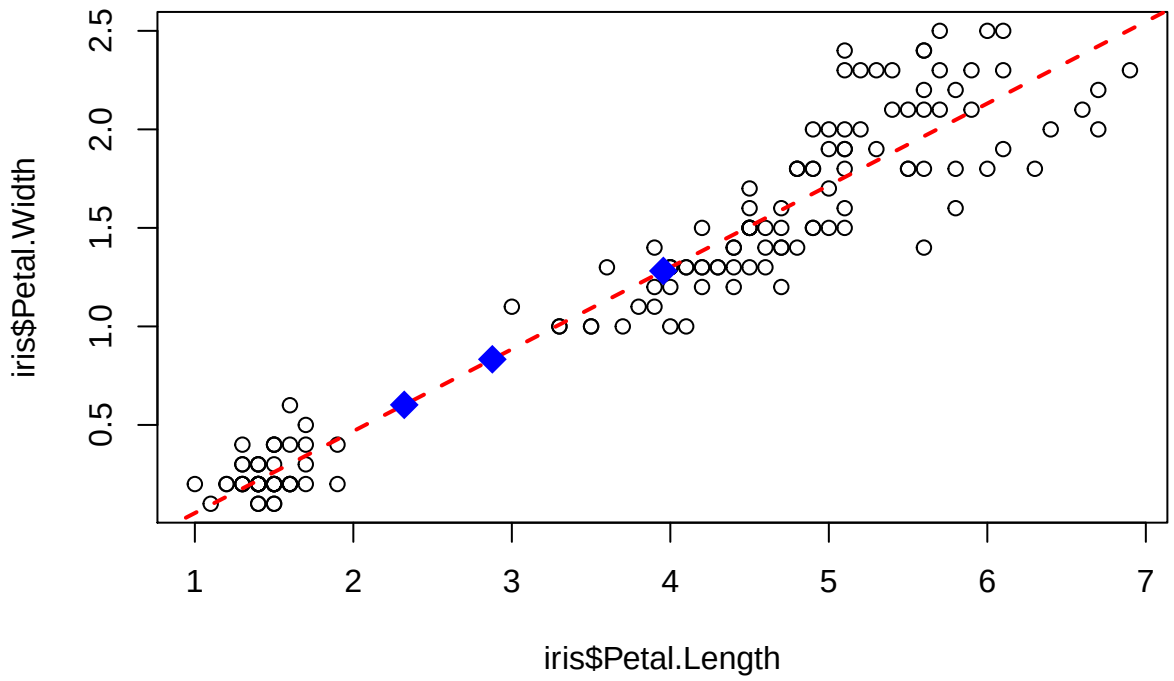
We can use the model fit object to pedict new data too. We just need a data frame of the new values with column names of the independent values the same as those in the original model:

```
new.petals <- data.frame(Petal.Length = runif(3, 2, 4))
new.petal.pred <- predict(petal.lm, new.petals)
new.petal.pred
```

```
        1         2         3
1.2815940 0.8330151 0.6019327
```

```
plot(iris$Petal.Length, iris$Petal.Width)
abline(petal.lm, col = "red", lwd = 2, lty = "dashed")
points(
  new.petals$Petal.Length,
  new.petal.pred,
  pch = 18,
  col = "blue",
```

```
  cex = 2
)
```

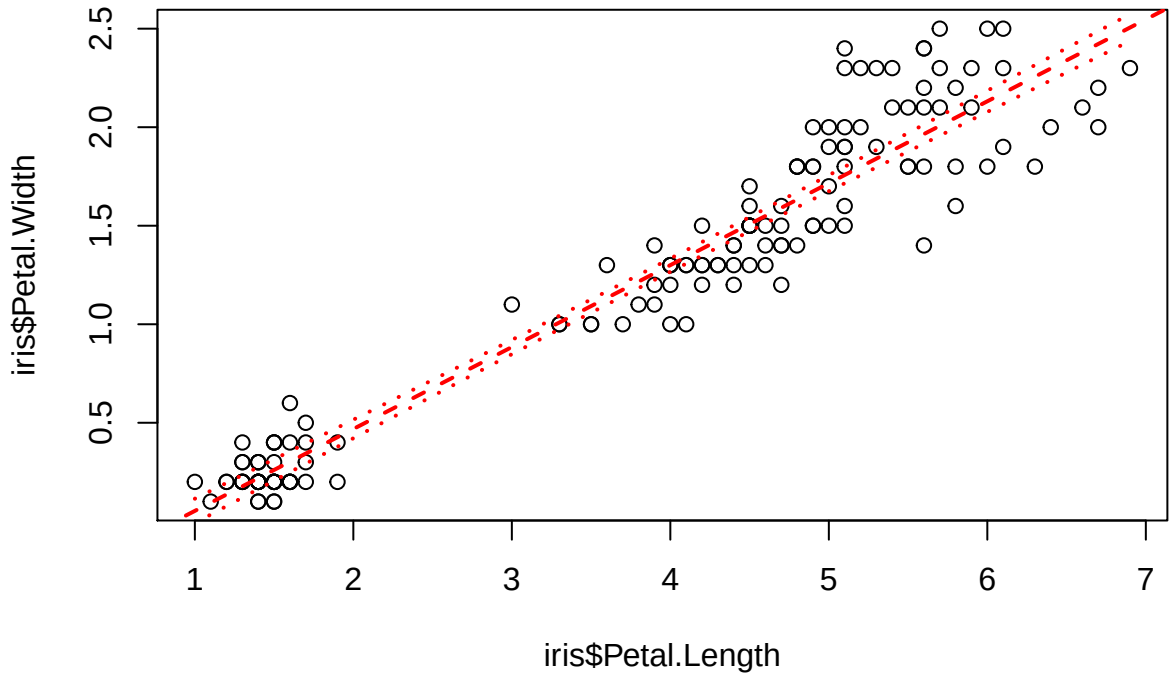


Predictions can also include confidence intervals:

```
# a data frame of 100 evenly spaced points from the max to min petal length
ci.df <- data.frame(
  Petal.Length = seq(min(iris$Petal.Length), max(iris$Petal.Length), length.out = 100)
)
new.petal.pred <- predict(petal.lm, ci.df, interval = "confidence")
head(new.petal.pred)
```

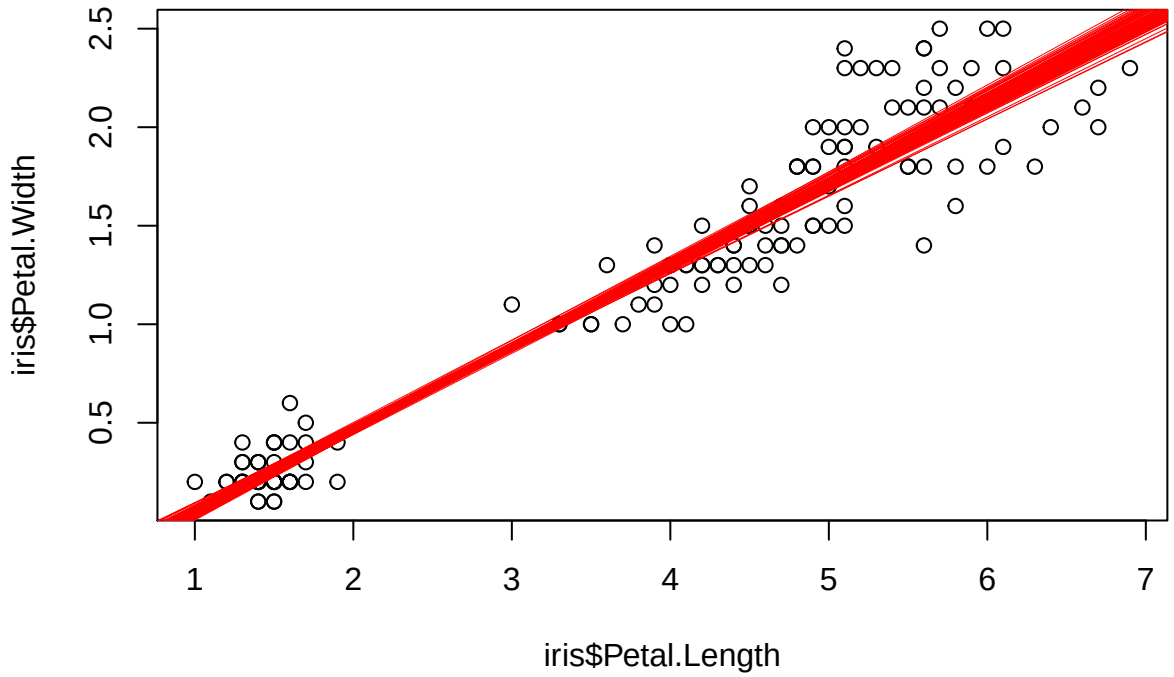
```
         fit          lwr       upr
1 0.05267990 -0.009267579 0.1146274
2 0.07745724  0.016458153 0.1384563
3 0.10223458  0.042177655 0.1622915
4 0.12701192  0.067890631 0.1861332
5 0.15178927  0.093596764 0.2099818
6 0.17656661  0.119295722 0.2338375
```

```
# plot points, fit, and confidence intervals
plot(iris$Petal.Length, iris$Petal.Width)
abline(petal.lm, col = "red", lwd = 2, lty = "dashed")
lines(ci.df$Petal.Length, new.petal.pred[, "lwr"], col = "red", lwd = 2, lty = "dotted")
lines(ci.df$Petal.Length, new.petal.pred[, "upr"], col = "red", lwd = 2, lty = "dotted")
```



Is that really the confidence interval? Most of the points lie outside of it! Yes, this is the confidence interval of the *fit*, not the variance around the fit (the residuals). We can prove that by bootstrapping the data and showing the distribution of the bootstrapped fits:

```
boot.lm <- lapply(1:100, function(i) {
  lm(
    Petal.Width ~ Petal.Length,
    data = iris[sample(1:nrow(iris), nrow(iris), T), ]
  )
})

plot(iris$Petal.Length, iris$Petal.Width)
for(x in boot.lm) abline(x, col = "red", lwd = 0.5)
```



Models can be built on categorical predictors as well. In this example we test whether or not petal length differs among species:

```
length.lm <- lm(Petal.Length ~ Species, iris)
summary(length.lm)
```

```
Call:
lm(formula = Petal.Length ~ Species, data = iris)

Residuals:
   Min     1Q Median     3Q    Max
-1.260 -0.258  0.038  0.240  1.348

Coefficients:
                 Estimate Std. Error t value Pr(>|t|)
(Intercept)       1.46200    0.06086   24.02   <2e-16 ***
Speciesversicolor 2.79800    0.08607   32.51   <2e-16 ***
Speciesvirginica  4.09000    0.08607   47.52   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.4303 on 147 degrees of freedom
Multiple R-squared:  0.9414,    Adjusted R-squared:  0.9406
F-statistic:  1180 on 2 and 147 DF,  p-value: < 2.2e-16
```

Note that the results list the dummy variables representing the levels of the categorical station predictor. Their estimated effects are expressed as being relative to the first level.

ANOVA

An analysis of variance (ANOVA) is related to the multi-category linear model and gets specified with the same formula using the `aov` function:

```r
length.aov <- aov(Petal.Length ~ Species, iris)
summary(length.aov)
```

```
             Df Sum Sq Mean Sq F value Pr(>F)
Species       2  437.1  218.55    1180 <2e-16 ***
Residuals   147   27.2    0.19
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
str(length.aov)
```

```
List of 13
 $ coefficients : Named num [1:3] 1.46 2.8 4.09
  ..- attr(*, "names")= chr [1:3] "(Intercept)" "Speciesversicolor" "Speciesvirginica"
 $ residuals    : Named num [1:150] -0.062 -0.062 -0.162 0.038 -0.062 ...
  ..- attr(*, "names")= chr [1:150] "1" "2" "3" "4" ...
 $ effects      : Named num [1:150] -46.026 4.347 20.45 0.058 -0.042 ...
  ..- attr(*, "names")= chr [1:150] "(Intercept)" "Speciesversicolor" "Speciesvirginica" "" ...
 $ rank         : int 3
 $ fitted.values: Named num [1:150] 1.46 1.46 1.46 1.46 1.46 ...
  ..- attr(*, "names")= chr [1:150] "1" "2" "3" "4" ...
 $ assign       : int [1:3] 0 1 1
 $ qr           :List of 5
  ..$ qr   : num [1:150, 1:3] -12.2474 0.0816 0.0816 0.0816 0.0816 ...
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : chr [1:150] "1" "2" "3" "4" ...
  .. .. ..$ : chr [1:3] "(Intercept)" "Speciesversicolor" "Speciesvirginica"
  .. ..- attr(*, "assign")= int [1:3] 0 1 1
  .. ..- attr(*, "contrasts")=List of 1
  .. .. ..$ Species: chr "contr.treatment"
  ..$ qraux: num [1:3] 1.08 1.05 1.09
  ..$ pivot: int [1:3] 1 2 3
  ..$ tol  : num 1e-07
  ..$ rank : int 3
  ..- attr(*, "class")= chr "qr"
 $ df.residual  : int 147
 $ contrasts    :List of 1
  ..$ Species: chr "contr.treatment"
 $ xlevels      :List of 1
  ..$ Species: chr [1:3] "setosa" "versicolor" "virginica"
 $ call         : language aov(formula = Petal.Length ~ Species, data = iris)
 $ terms        :Classes 'terms', 'formula'  language Petal.Length ~ Species
  .. ..- attr(*, "variables")= language list(Petal.Length, Species)
  .. ..- attr(*, "factors")= int [1:2, 1] 0 1
  .. .. ..- attr(*, "dimnames")=List of 2
  .. .. .. ..$ : chr [1:2] "Petal.Length" "Species"
  .. .. .. ..$ : chr "Species"
  .. ..- attr(*, "term.labels")= chr "Species"
  .. ..- attr(*, "order")= int 1
  .. ..- attr(*, "intercept")= int 1
  .. ..- attr(*, "response")= int 1
  .. ..- attr(*, ".Environment")=<environment: R_GlobalEnv>
  .. ..- attr(*, "predvars")= language list(Petal.Length, Species)
  .. ..- attr(*, "dataClasses")= Named chr [1:2] "numeric" "factor"
```

```
  .. .. ..- attr(*, "names")= chr [1:2] "Petal.Length" "Species"
 $ model        :'data.frame':  150 obs. of  2 variables:
  ..$ Petal.Length: num [1:150] 1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
  ..$ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
  ..- attr(*, "terms")=Classes 'terms', 'formula'  language Petal.Length ~ Species
  .. .. ..- attr(*, "variables")= language list(Petal.Length, Species)
  .. .. ..- attr(*, "factors")= int [1:2, 1] 0 1
  .. .. .. ..- attr(*, "dimnames")=List of 2
  .. .. .. .. ..$ : chr [1:2] "Petal.Length" "Species"
  .. .. .. .. ..$ : chr "Species"
  .. .. ..- attr(*, "term.labels")= chr "Species"
  .. .. ..- attr(*, "order")= int 1
  .. .. ..- attr(*, "intercept")= int 1
  .. .. ..- attr(*, "response")= int 1
  .. .. ..- attr(*, ".Environment")=<environment: R_GlobalEnv>
  .. .. ..- attr(*, "predvars")= language list(Petal.Length, Species)
  .. .. ..- attr(*, "dataClasses")= Named chr [1:2] "numeric" "factor"
  .. .. .. ..- attr(*, "names")= chr [1:2] "Petal.Length" "Species"
 - attr(*, "class")= chr [1:2] "aov" "lm"
```

The analysis of variance table can also be computed from an `lm` object using `anova`:

```
anova(length.lm)
```

```
Analysis of Variance Table

Response: Petal.Length
           Df Sum Sq Mean Sq F value    Pr(>F)
Species     2 437.10 218.551  1180.2 < 2.2e-16 ***
Residuals 147  27.22   0.185
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Differences between levels of the predictor can be tested with the `TukeyHSD()` function:

```
TukeyHSD(length.aov)
```

```
  Tukey multiple comparisons of means
    95% family-wise confidence level

Fit: aov(formula = Petal.Length ~ Species, data = iris)

$Species
                      diff     lwr     upr p adj
versicolor-setosa    2.798 2.59422 3.00178     0
virginica-setosa     4.090 3.88622 4.29378     0
virginica-versicolor 1.292 1.08822 1.49578     0
```

Permutation tests

If we don't want to rely on canned parametric assessments of differences among groups, we can construct the null distributions by randomly permuting group assignments and comparing the distribution of the test statistic with the observed value. As an example here's the observed t-statistic for our simulated length measurements from the beginning:

```
spp.ttest$statistic
```

```
        t
-1.572637
```

The t-statistic for one random permutation of species can be calculated by permuting the species column of the data frame and running the t-test again:

```
perm.spp <- sample(length.df$spp)
t.test(length.df$length ~ perm.spp)$statistic
```

```
        t
0.2401946
```

We want to run this a number of times and create a vector of the permuted t-statistics. Let's use `sapply` to create a matrix of permutations and then `apply` to walk through that matrix and run the t.test for each permutation using an anonymous function:

```
# matrix of permuted species designations
perm.spp.mat <- sapply(1:1000, function(i) sample(length.df$spp))
str(perm.spp.mat)
```

```
 num [1:216, 1:1000] 1 2 1 2 2 2 1 2 2 2 ...
```

```
# vector of t-statistics for each permutation
perm.t <- apply(perm.spp.mat, 2, function(spp) {
  t.test(length.df$length ~ spp)$statistic
})
str(perm.t)
```

```
 num [1:1000] -0.89599 0.27409 0.71318 0.00269 1.63731 ...
```

If we don't want to save the permutations, we can both permute and run the t-test with `sapply`:

```
perm.t <- sapply(1:1000, function(i) {
  perm.spp <- sample(length.df$spp)
  t.test(length.df$length ~ perm.spp)$statistic
})
str(perm.t)
```

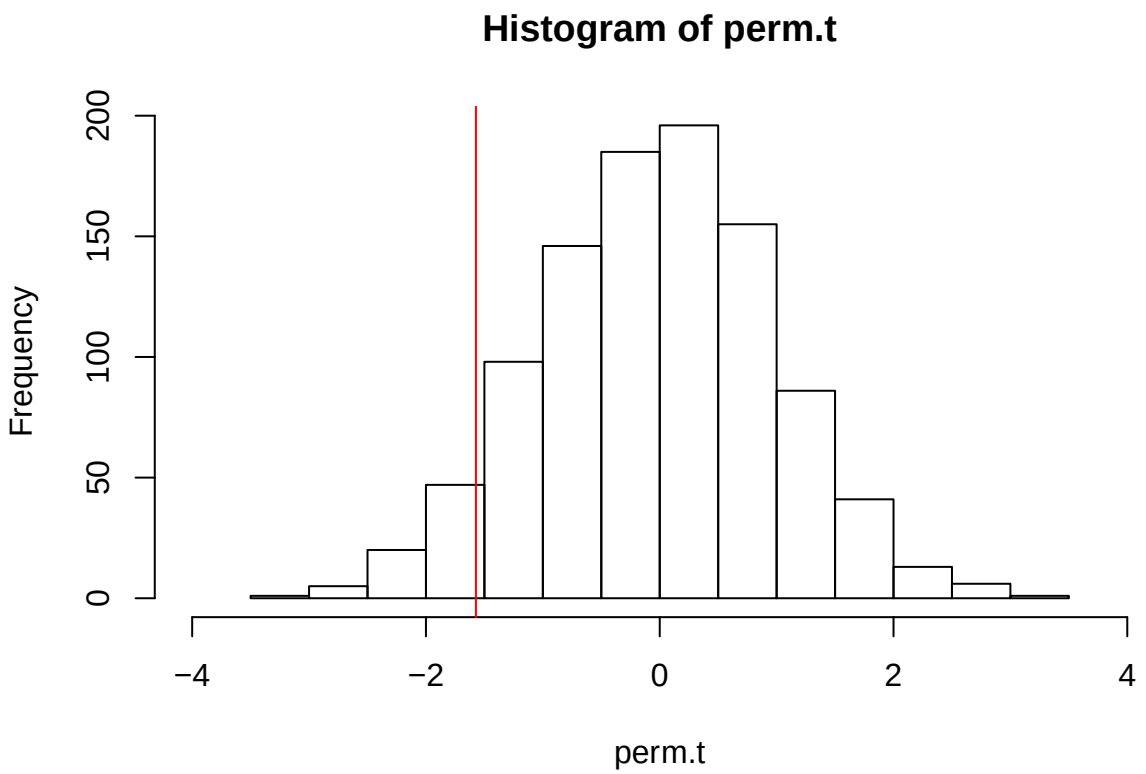
```
 Named num [1:1000] -0.76627 -0.31312 1.53265 0.08491 -0.00443 ...
 - attr(*, "names")= chr [1:1000] "t" "t" "t" "t" ...
```

Now let's summarize the results and compare the distribution to the observed value.

```
summary(perm.t)
```

```
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
-3.07313 -0.71061 -0.00845 -0.02241  0.64964  3.35464
```

```
obs.t <- spp.ttest$statistic

# a histogram of the permutation t-distribution
hist(perm.t, xlim = range(pretty(c(perm.t, obs.t))))
abline(v = obs.t, col = "red")
```




```
# what percentage of the distribution is >= the observed?
mean(perm.t >= obs.t)
```

```
[1] 0.937
```

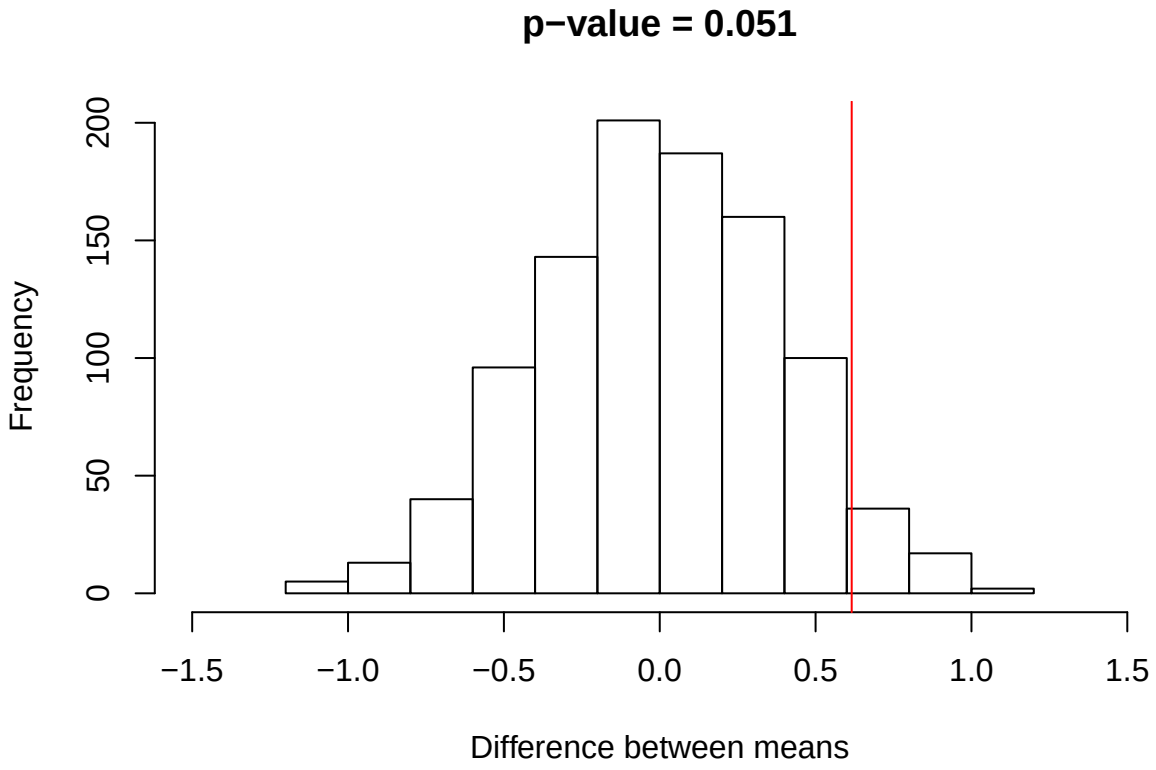
```
# compared to t-test p-value:
spp.ttest$p.value
```

```
[1] 0.1173081
```

A more proper permutation t-test is to create a null distribution of the value we're interested in - the difference betwee means, rather than the distribution of the test statistic. Let's make our lives easier by creating a function that returns the difference among means given a data frame like `length.df`, then running the permutation test with this function.

```
meanDiff <- function(x) {
  # calculate mean length for both groups
  spp.mean <- tapply(x$length, x$spp, mean)
  # return difference
  diff(spp.mean)
}

# the observed difference
obs.diff <- meanDiff(length.df)

# collect vector of differences from permutations
perm.df <- length.df # a copy that we'll be modifying
perm.diff <- sapply(1:1000, function(i) {
  perm.df$spp <- sample(perm.df$spp)
  meanDiff(perm.df)
})
```

```r
# calculate p-value based on sign of observed difference
p.value <- if(obs.diff < 0) {
  mean(perm.diff <= obs.diff)
} else {
  mean(perm.diff >= obs.diff)
}

# show relationship of difference to null distribution
hist(
  perm.diff,
  xlab = "Difference between means",
  xlim = range(pretty(c(perm.diff, obs.diff))),
  main = paste("p-value =", round(p.value, 3))
)
abline(v = obs.diff, col = "red")
```



Pairwise analyses

One way to do pairwise analyses in R is to use the function `combn` which will generate all possible combinations of length `m` of elements of a vector. For example, here's all possible combinations of 2 species:

```r
combn(levels(iris$Species), 2)
```

```
     [,1]         [,2]        [,3]
[1,] "setosa"     "setosa"    "versicolor"
[2,] "versicolor" "virginica" "virginica"
```

The `combn` function will also supply each combination to a function of your choosing, for example, we can paste the names together:

```
combn(levels(iris$Species), 2, paste, collapse = " v. ")
```

```
[1] "setosa v. versicolor"    "setosa v. virginica"
[3] "versicolor v. virginica"
```

. . . or we can use the value delivered by the function to extract some data and calculate a value between each pair, like the difference in mean petal length:

```
combn(levels(iris$Species), 2, function(x) {
  x.df <- droplevels(iris[iris$Species %in% x, ])
  petal.length.means <- tapply(x.df$Petal.Length, x.df$Species, mean)
  diff(petal.length.means)
})
```

```
[1] 2.798 4.090 1.292
```

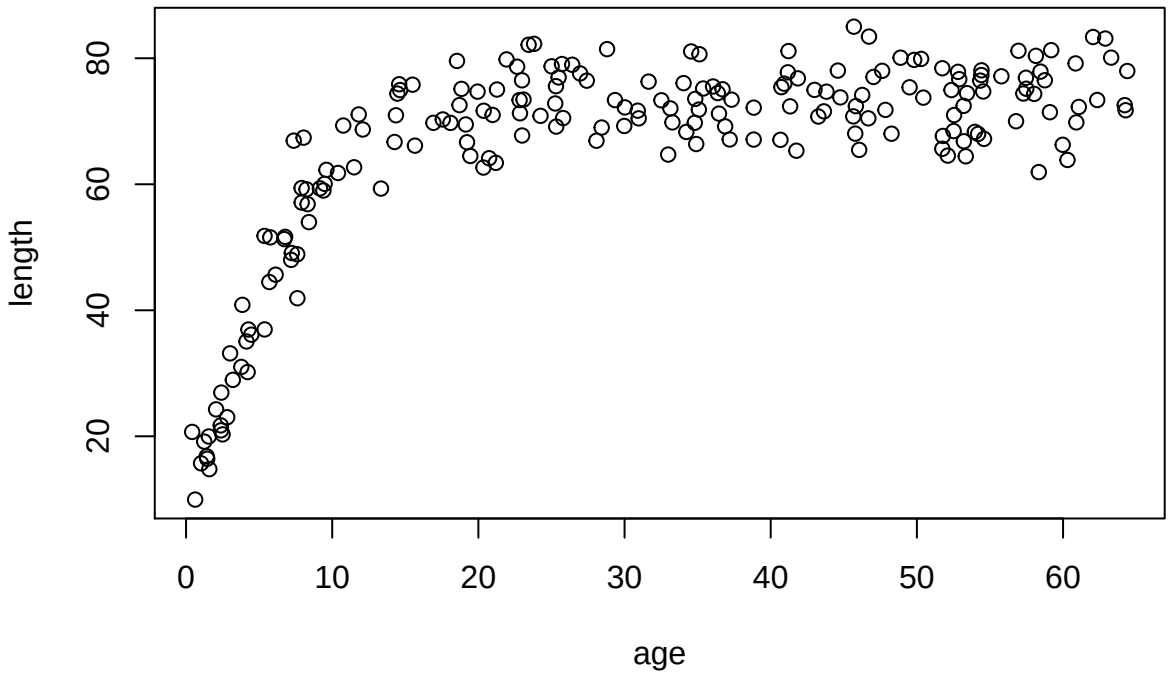
Non-linear models

To illustrate non-linear model fitting, we'll first create a function to simulate growth data (length ~ age) based on a Gompertz curve. The Gompertz function is $length = L_0 \cdot e^{k(1-e^{-g \cdot age})}$, where L_0 is the length at birth (LAB). Here's the function to create simulated growth data:

```
# age.range - a two element vector giving the minimum and maximum ages
# lab - the length at birth
# k, g - displacement and rate parameters
# std.dev - standard deviation for the error term
# sample.size - number of points to simulate

sim.growth.func <- function(age.range, lab, k, g, std.dev, sample.size) {
  # Generate some random ages between min and max of age.range
  ages <- runif(sample.size, age.range[1], age.range[2])
  # Calculate the expected length for those ages from the Gompertz equation
  expected.length <- lab * exp(k * (1 - exp(-g * ages)))
  # Add some error to the lengths and return the named array
  length.err <- rnorm(sample.size, 0, std.dev)
  as.data.frame(cbind(age = ages, length = expected.length + length.err))
}
```

With this function, we can now simulate some growth data:

```
growth.df <- sim.growth.func(
  age.range = c(0, 65),
  lab = 10,
  k = 2,
  g = 0.25,
  std.dev = 5,
  sample.size = 200
)
plot(length ~ age, growth.df)
```



Now let's use nonlinear least squares to estimate the parameters from this simulated data. We can do that with the `nls` function, which behaves very similarly to `lm`. The main difference is that we need to supply initial values, which are specified in the third argument, `start`. These values should be chosen carefully so as to ensure convergence.

```
gr.form <- length ~ lab * exp(k * (1 - exp(-g * age)))
# starting values for k and g are too far off for default number of iterations
gr.nls <- nls(gr.form, growth.df, start = c(lab = 15, k = 10, g = 10))
```

```
Error in nls(gr.form, growth.df, start = c(lab = 15, k = 10, g = 10)): singular gradient
```

```
# this should work
gr.nls <- nls(gr.form, growth.df, start = c(lab = 15, k = 1, g = 0.5))
print(gr.nls)
```

```
Nonlinear regression model
  model: length ~ lab * exp(k * (1 - exp(-g * age)))
   data: growth.df
  lab     k     g
9.494 2.048 0.253
 residual sum-of-squares: 4950

Number of iterations to convergence: 6
Achieved convergence tolerance: 1.046e-06
```

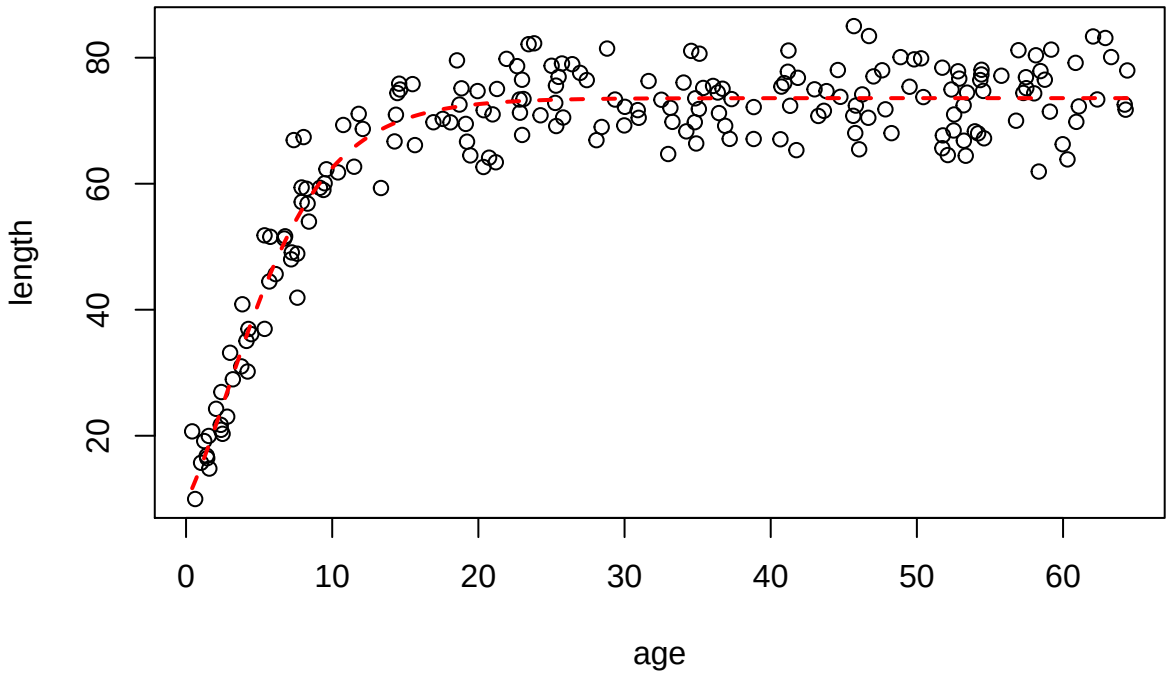
...and here are the estimated coefficients:

```
gr.coef <- coef(gr.nls)
gr.coef
```

```
      lab         k         g
9.4936600 2.0477619 0.2530218
```

We'll plot the fitted curve:

```
grow.fit <- data.frame(
  age = seq(
    min(growth.df$age),
    max(growth.df$age),
    length.out = 1000
  )
)
grow.fit$length <- predict(gr.nls, grow.fit)
plot(length ~ age, growth.df)
lines(grow.fit$age, grow.fit$length, col = "red", lwd = 2, lty = "dashed")
```



Tidy Data: tidyverse

Piping

```
# method one of doing three steps (sequential)
x <- runif(100)
x.q <- quantile(x, c(0.025, 0.975))
x.q.diff.1 <- diff(x.q)

# method two (nested)
x.q.diff.2 <- diff(quantile(runif(100), c(0.025, 0.975)))
```

Piping (from package `magrittr`) uses the `%>%` operator

```
library(magrittr)
runif(10)
```

```
 [1] 0.6765419 0.1349971 0.8854835 0.9545984 0.3007419 0.9860582 0.2358094
 [8] 0.8488136 0.1916240 0.5379602
```

```
10 %>% runif()
```

```
 [1] 0.23103636 0.68050336 0.67362630 0.54883845 0.80439622 0.33410543
 [7] 0.01638260 0.11706423 0.08348733 0.25814155
```

```
# no parentheses needed if left side is all that is going into function
10 %>% runif
```

```
 [1] 0.5617633 0.6100165 0.9496892 0.3974650 0.4482829 0.1384789 0.4318470
 [8] 0.4935638 0.9157182 0.7738272
```

```
# using arguments
10 %>% runif(100, 200)
```

```
 [1] 184.8563 199.8106 106.5240 133.9597 188.5539 160.0892 168.1523 191.4025
 [9] 132.8358 131.8538
```

```
# pipe to second argument (must name arguments)
100 %>% runif(n = 5, max = 200)
```

```
[1] 197.4230 116.4569 109.3535 156.3652 187.4226
```

```
# vs...
100 %>% runif(5, 200)
```

```
  [1]  97.798754 153.995685 151.435934 162.938124  58.314364   5.825593
  [7] 157.002141  56.304297 146.520167 190.177507 197.199791  97.447649
 [13]  48.884789 135.857086 175.398141  65.173079 129.247353  26.435516
 [19] 123.277937 143.875545 116.221208 166.423692  58.324623 176.548831
 [25]  78.294313 122.717485 114.512507 109.871890 184.383058  83.151711
 [31]   7.078127  70.043553  50.859049 143.124879 149.983387  65.957638
 [37] 115.794598  80.315897 121.380334  15.928588 117.527628 125.842998
 [43] 190.204206 125.880306  91.539547  88.472337 137.827639  10.992599
 [49] 121.793518  56.046133 138.810668  78.625437 158.476739 101.087021
 [55] 118.349536  60.365085 145.123838 176.617404 148.467315  23.614938
 [61]  66.284105  45.894007  93.951598  81.158081  26.608829 117.319235
 [67] 121.104426 119.967168  68.079918 127.038410  69.062198  37.226137
 [73]  32.392585 190.739701  84.165794  60.346598  65.384097 145.994459
 [79] 183.269396  17.880766 179.213215  45.367236 108.280274 122.411083
 [85]  42.220379 121.537639 175.593617  17.909841 100.691673 167.468020
 [91]  46.392299  71.913170 147.996639 121.599319 122.104449  44.534038
 [97]  93.040730  28.365526  60.100147 147.181503
```

pipe version of first example

```
q.diff.pipe <- 100 %>%
  runif %>%
  quantile(c(0.025, 0.975)) %>%
  diff
```

dplyr

filter and select

```
library(tidyverse)
```

```
-- Attaching packages ------------------------------------- tidyverse 1.3.0 --
```

```
v ggplot2 3.2.1     v purrr   0.3.3
v tibble  2.1.3     v dplyr   0.8.4
```

```
v tidyr   1.0.2     v stringr 1.4.0
v readr   1.3.1     v forcats 0.4.0

-- Conflicts ---------------------------------------- tidyverse_conflicts() --
x tidyr::extract()   masks magrittr::extract()
x dplyr::filter()    masks stats::filter()
x dplyr::lag()       masks stats::lag()
x purrr::set_names() masks magrittr::set_names()
```

```
# base R indexing to select males
#starwars[starwars$gender == "male", ]
#subset(starwars, gender == "male")

# dplyr way - filter
filter(starwars, gender == "male")
```

```
# A tibble: 62 x 13
   name  height  mass hair_color skin_color eye_color birth_year gender
   <chr>  <int> <dbl> <chr>      <chr>      <chr>          <dbl> <chr>
 1 Luke~    172    77 blond      fair       blue            19   male
 2 Dart~    202   136 none       white      yellow          41.9 male
 3 Owen~    178   120 brown, gr~ light      blue            52   male
 4 Bigg~    183    84 black      light      brown           24   male
 5 Obi-~    182    77 auburn, w~ fair       blue-gray       57   male
 6 Anak~    188    84 blond      fair       blue            41.9 male
 7 Wilh~    180    NA auburn, g~ fair       blue            64   male
 8 Chew~    228   112 brown      unknown    blue           200   male
 9 Han ~    180    80 brown      fair       brown           29   male
10 Gree~    173    74 <NA>       green      black           44   male
# ... with 52 more rows, and 5 more variables: homeworld <chr>, species <chr>,
#   films <list>, vehicles <list>, starships <list>
```

```
# pipeline version
starwars %>%
  filter(gender == "male" & height > 190)
```

```
# A tibble: 20 x 13
   name  height  mass hair_color skin_color eye_color birth_year gender
   <chr>  <int> <dbl> <chr>      <chr>      <chr>          <dbl> <chr>
 1 Dart~    202   136 none       white      yellow          41.9 male
 2 Chew~    228   112 brown      unknown    blue           200   male
 3 Qui-~    193    89 brown      fair       blue            92   male
 4 Nute~    191    90 none       mottled g~ red             NA   male
 5 Jar ~    196    66 none       orange     orange          52   male
 6 Roos~    224    82 none       grey       orange          NA   male
 7 Rugo~    206    NA none       green      orange          NA   male
 8 Ki-A~    198    82 white      pale       yellow          92   male
 9 Kit ~    196    87 none       green      black           NA   male
10 Yara~    264    NA none       white      yellow          NA   male
11 Mas ~    196    NA none       blue       blue            NA   male
12 Dooku    193    80 white      fair       brown          102   male
13 Bail~    191    NA black      tan        brown           67   male
14 Dext~    198   102 none       brown      yellow          NA   male
15 Lama~    229    88 none       grey       black           NA   male
16 Wat ~    193    48 none       green, gr~ unknown         NA   male
17 San ~    191    NA none       grey       gold            NA   male
```

```
18 Grie~    216   159 none       brown, wh~ green, y~       NA   male
19 Tarf~    234   136 brown      brown      blue            NA   male
20 Tion~    206    80 none       grey       black           NA   male
# ... with 5 more variables: homeworld <chr>, species <chr>, films <list>,
#   vehicles <list>, starships <list>
```

```
# "select" columns to return
select(starwars, name, height, mass, gender)
```

```
# A tibble: 87 x 4
   name               height  mass gender
   <chr>               <int> <dbl> <chr>
 1 Luke Skywalker        172    77 male
 2 C-3PO                 167    75 <NA>
 3 R2-D2                  96    32 <NA>
 4 Darth Vader           202   136 male
 5 Leia Organa           150    49 female
 6 Owen Lars             178   120 male
 7 Beru Whitesun lars    165    75 female
 8 R5-D4                  97    32 <NA>
 9 Biggs Darklighter     183    84 male
10 Obi-Wan Kenobi        182    77 male
# ... with 77 more rows
```

```
select(starwars, height, gender, name, mass)
```

```
# A tibble: 87 x 4
   height gender name               mass
    <int> <chr>  <chr>             <dbl>
 1    172 male   Luke Skywalker       77
 2    167 <NA>   C-3PO                75
 3     96 <NA>   R2-D2                32
 4    202 male   Darth Vader         136
 5    150 female Leia Organa          49
 6    178 male   Owen Lars           120
 7    165 female Beru Whitesun lars   75
 8     97 <NA>   R5-D4                32
 9    183 male   Biggs Darklighter    84
10    182 male   Obi-Wan Kenobi       77
# ... with 77 more rows
```

```
# extend pipeline above
starwars %>%
  filter(gender == "male" & height > 190) %>%
  select(name, height, mass)
```

```
# A tibble: 20 x 3
   name               height  mass
   <chr>               <int> <dbl>
 1 Darth Vader           202   136
 2 Chewbacca             228   112
 3 Qui-Gon Jinn          193    89
 4 Nute Gunray           191    90
 5 Jar Jar Binks         196    66
 6 Roos Tarpals          224    82
 7 Rugor Nass            206    NA
```

```
 8 Ki-Adi-Mundi           198    82
 9 Kit Fisto              196    87
10 Yarael Poof            264    NA
11 Mas Amedda             196    NA
12 Dooku                  193    80
13 Bail Prestor Organa    191    NA
14 Dexter Jettster        198   102
15 Lama Su                229    88
16 Wat Tambor             193    48
17 San Hill               191    NA
18 Grievous               216   159
19 Tarfful                234   136
20 Tion Medon             206    80
```

```r
# helper functions for select

# select range of columns
starwars %>%
  filter(gender == "male" & height > 190) %>%
  select(eye_color:homeworld)
```

```
# A tibble: 20 x 4
   eye_color     birth_year gender homeworld
   <chr>              <dbl> <chr>  <chr>
 1 yellow              41.9 male   Tatooine
 2 blue               200   male   Kashyyyk
 3 blue                92   male   <NA>
 4 red                 NA   male   Cato Neimoidia
 5 orange              52   male   Naboo
 6 orange              NA   male   Naboo
 7 orange              NA   male   Naboo
 8 yellow              92   male   Cerea
 9 black               NA   male   Glee Anselm
10 yellow              NA   male   Quermia
11 blue                NA   male   Champala
12 brown              102   male   Serenno
13 brown               67   male   Alderaan
14 yellow              NA   male   Ojom
15 black               NA   male   Kamino
16 unknown             NA   male   Skako
17 gold                NA   male   Muunilinst
18 green, yellow       NA   male   Kalee
19 blue                NA   male   Kashyyyk
20 black               NA   male   Utapau
```

```r
# select columns that start with string
starwars %>%
  filter(gender == "male" & height > 190) %>%
  select(starts_with("h"))
```

```
# A tibble: 20 x 3
   height hair_color homeworld
    <int> <chr>      <chr>
 1    202 none       Tatooine
 2    228 brown      Kashyyyk
 3    193 brown      <NA>
```

```
 4    191 none       Cato Neimoidia
 5    196 none       Naboo
 6    224 none       Naboo
 7    206 none       Naboo
 8    198 white      Cerea
 9    196 none       Glee Anselm
10    264 none       Quermia
11    196 none       Champala
12    193 white      Serenno
13    191 black      Alderaan
14    198 none       Ojom
15    229 none       Kamino
16    193 none       Skako
17    191 none       Muunilinst
18    216 none       Kalee
19    234 brown      Kashyyyk
20    206 none       Utapau
```

```
# select columns that contain a string
starwars %>%
  filter(gender == "male" & height > 190) %>%
  select(contains("color"))
```

```
# A tibble: 20 x 3
   hair_color skin_color    eye_color
   <chr>      <chr>         <chr>
 1 none       white         yellow
 2 brown      unknown       blue
 3 brown      fair          blue
 4 none       mottled green red
 5 none       orange        orange
 6 none       grey          orange
 7 none       green         orange
 8 white      pale          yellow
 9 none       green         black
10 none       white         yellow
11 none       blue          blue
12 white      fair          brown
13 black      tan           brown
14 none       brown         yellow
15 none       grey          black
16 none       green, grey   unknown
17 none       grey          gold
18 none       brown, white  green, yellow
19 brown      brown         blue
20 none       grey          black
```

```
# select columns excluding certain ones
starwars %>%
  filter(gender == "male" & height > 190) %>%
  select(-name, -gender, -height)
```

```
# A tibble: 20 x 10
    mass hair_color skin_color eye_color birth_year homeworld species films
   <dbl> <chr>      <chr>      <chr>          <dbl> <chr>     <chr>   <lis>
 1   136 none       white      yellow          41.9 Tatooine  Human   <chr~
```

```
 2   112 brown      unknown    blue           200   Kashyyyk  Wookiee <chr~
 3    89 brown      fair       blue            92   <NA>      Human   <chr~
 4    90 none       mottled g~ red             NA   Cato Nei~ Neimod~ <chr~
 5    66 none       orange     orange          52   Naboo     Gungan  <chr~
 6    82 none       grey       orange          NA   Naboo     Gungan  <chr~
 7    NA none       green      orange          NA   Naboo     Gungan  <chr~
 8    82 white      pale       yellow          92   Cerea     Cerean  <chr~
 9    87 none       green      black           NA   Glee Ans~ Nautol~ <chr~
10    NA none       white      yellow          NA   Quermia   Quermi~ <chr~
11    NA none       blue       blue            NA   Champala  Chagri~ <chr~
12    80 white      fair       brown          102   Serenno   Human   <chr~
13    NA black      tan        brown           67   Alderaan  Human   <chr~
14   102 none       brown      yellow          NA   Ojom      Besali~ <chr~
15    88 none       grey       black           NA   Kamino    Kamino~ <chr~
16    48 none       green, gr~ unknown         NA   Skako     Skakoan <chr~
17    NA none       grey       gold            NA   Muunilin~ Muun    <chr~
18   159 none       brown, wh~ green, y~       NA   Kalee     Kaleesh <chr~
19   136 brown      brown      blue            NA   Kashyyyk  Wookiee <chr~
20    80 none       grey       black           NA   Utapau    Pau'an  <chr~
# ... with 2 more variables: vehicles <list>, starships <list>
```

arrange to sort data

```
# base R sorting a data.frame
starwars[order(starwars$species, starwars$height), ]
```

```
# A tibble: 87 x 13
   name  height  mass hair_color skin_color eye_color birth_year gender
   <chr>  <int> <dbl> <chr>      <chr>      <chr>          <dbl> <chr>
 1 Ratt~     79    15 none       grey, blue unknown           NA male
 2 Dext~    198   102 none       brown      yellow            NA male
 3 Ki-A~    198    82 white      pale       yellow            92 male
 4 Mas ~    196    NA none       blue       blue              NA male
 5 Zam ~    168    55 blonde     fair, gre~ yellow            NA female
 6 R2-D2     96    32 <NA>       white, bl~ red               33 <NA>
 7 R5-D4     97    32 <NA>       white, red red               NA <NA>
 8 C-3PO    167    75 <NA>       gold       yellow           112 <NA>
 9 IG-88    200   140 none       metal      red               15 none
10 BB8       NA    NA none       none       black             NA none
# ... with 77 more rows, and 5 more variables: homeworld <chr>, species <chr>,
#   films <list>, vehicles <list>, starships <list>
```

```
# arrange
starwars %>%
  arrange(species, desc(height)) %>%
  select(name, height, species)
```

```
# A tibble: 87 x 3
   name            height species
   <chr>            <int> <chr>
 1 Ratts Tyerell       79 Aleena
 2 Dexter Jettster    198 Besalisk
 3 Ki-Adi-Mundi       198 Cerean
 4 Mas Amedda         196 Chagrian
 5 Zam Wesell         168 Clawdite
 6 IG-88              200 Droid
```

```
 7 C-3PO              167 Droid
 8 R5-D4               97 Droid
 9 R2-D2               96 Droid
10 BB8                 NA Droid
# ... with 77 more rows
```

new columns

```
sw <- starwars %>%
  mutate(
    height.m = height / 100,
    bmi = mass / height.m ^ 2
  )

# takes place of
# sw <- starwars
# sw$height.m <- sw$height / 100
# sw$bmi <- sw$mass / sw$height.m ^ 2
```

change name of column

```
sw <- starwars %>%
  rename(handle = "name")
colnames(starwars)
```

```
 [1] "name"       "height"     "mass"       "hair_color" "skin_color"
 [6] "eye_color"  "birth_year" "gender"     "homeworld"  "species"
[11] "films"      "vehicles"   "starships"
```

```
colnames(sw)
```

```
 [1] "handle"     "height"     "mass"       "hair_color" "skin_color"
 [6] "eye_color"  "birth_year" "gender"     "homeworld"  "species"
[11] "films"      "vehicles"   "starships"
```

create new column and drop all others

```
sw <- starwars %>%
  transmute(
    name = name,
    height.m = height / 100,
    bmi = mass / height.m ^ 2
  )
sw
```

```
# A tibble: 87 x 3
   name               height.m   bmi
   <chr>                 <dbl> <dbl>
 1 Luke Skywalker         1.72  26.0
 2 C-3PO                  1.67  26.9
 3 R2-D2                  0.96  34.7
 4 Darth Vader            2.02  33.3
 5 Leia Organa            1.5   21.8
 6 Owen Lars              1.78  37.9
 7 Beru Whitesun lars     1.65  27.5
 8 R5-D4                  0.97  34.0
 9 Biggs Darklighter      1.83  25.1
10 Obi-Wan Kenobi         1.82  23.2
```

```
# ... with 77 more rows
```

```
# same as
sw <- starwars %>%
  mutate(
    height.m = height / 100,
    bmi = mass / height.m ^ 2
  ) %>%
  select(height.m, bmi)
sw
```

```
# A tibble: 87 x 2
   height.m   bmi
      <dbl> <dbl>
 1     1.72  26.0
 2     1.67  26.9
 3     0.96  34.7
 4     2.02  33.3
 5     1.5   21.8
 6     1.78  37.9
 7     1.65  27.5
 8     0.97  34.0
 9     1.83  25.1
10     1.82  23.2
# ... with 77 more rows
```

complete data set (no missing data)

```
# in base R
#sw.complete <- starwars[complete.cases(starwars), ]

sw.complete <- starwars %>%
  select(-(films:starships), -mass) %>%
  filter(complete.cases(.))

nrow(starwars)
```

```
[1] 87
```

```
nrow(sw.complete)
```

```
[1] 35
```

```
sw.complete
```

```
# A tibble: 35 x 9
   name  height hair_color skin_color eye_color birth_year gender homeworld
   <chr>  <int> <chr>      <chr>      <chr>          <dbl> <chr>  <chr>
 1 Luke~    172 blond      fair       blue            19   male   Tatooine
 2 Dart~    202 none       white      yellow          41.9 male   Tatooine
 3 Leia~    150 brown      light      brown           19   female Alderaan
 4 Owen~    178 brown, gr~ light      blue            52   male   Tatooine
 5 Beru~    165 brown      light      blue            47   female Tatooine
 6 Bigg~    183 black      light      brown           24   male   Tatooine
 7 Obi-~    182 auburn, w~ fair       blue-gray       57   male   Stewjon
 8 Anak~    188 blond      fair       blue            41.9 male   Tatooine
 9 Wilh~    180 auburn, g~ fair       blue            64   male   Eriadu
10 Chew~    228 brown      unknown    blue           200   male   Kashyyyk
```

```
# ... with 25 more rows, and 1 more variable: species <chr>
```

removing duplicates

```
# what are the observed combinations of gender and species
starwars %>%
  select(gender, species) %>%
  distinct() %>%
  arrange(species, gender)
```

```
# A tibble: 43 x 2
   gender species
   <chr>  <chr>
 1 male   Aleena
 2 male   Besalisk
 3 male   Cerean
 4 male   Chagrian
 5 female Clawdite
 6 none   Droid
 7 <NA>   Droid
 8 male   Dug
 9 male   Ewok
10 male   Geonosian
# ... with 33 more rows
```

select random rows

```
# without replacement
starwars %>%
  sample_n(10)
```

```
# A tibble: 10 x 13
   name  height  mass hair_color skin_color eye_color birth_year gender
   <chr>  <int> <dbl> <chr>      <chr>      <chr>          <dbl> <chr>
 1 Boba~    183  78.2 black      fair       brown           31.5 male
 2 Saes~    188  NA   none       pale       orange          NA   male
 3 Greg~    185  85   black      dark       brown           NA   male
 4 Capt~     NA  NA   unknown    unknown    unknown         NA   female
 5 Nute~    191  90   none       mottled g~ red             NA   male
 6 Anak~    188  84   blond      fair       blue            41.9 male
 7 San ~    191  NA   none       grey       gold            NA   male
 8 Wilh~    180  NA   auburn, g~ fair       blue            64   male
 9 R2-D2     96  32   <NA>       white, bl~ red             33   <NA>
10 Dud ~     94  45   none       blue, grey yellow          NA   male
# ... with 5 more variables: homeworld <chr>, species <chr>, films <list>,
#   vehicles <list>, starships <list>
```

```
# with replacement
starwars %>%
  sample_n(10, weight = sample(1:10, nrow(.), replace = T))
```

```
# A tibble: 10 x 13
   name  height  mass hair_color skin_color eye_color birth_year gender
   <chr>  <int> <dbl> <chr>      <chr>      <chr>          <dbl> <chr>
 1 Greg~    185    85 black      dark       brown             NA male
 2 Ratt~     79    15 none       grey, blue unknown           NA male
 3 R5-D4     97    32 <NA>       white, red red               NA <NA>
```

```
 4 Jabb~    175  1358 <NA>       green-tan~ orange           600 herma~
 5 Mon ~    150    NA auburn     fair       blue              48 female
 6 Sebu~    112    40 none       grey, red  orange            NA male
 7 Zam ~    168    55 blonde     fair, gre~ yellow            NA female
 8 Bigg~    183    84 black      light      brown             24 male
 9 Luke~    172    77 blond      fair       blue              19 male
10 Dormé    165    NA brown      light      brown             NA female
# ... with 5 more variables: homeworld <chr>, species <chr>, films <list>,
#   vehicles <list>, starships <list>
```

group_by

```
sw <- starwars %>%
  group_by(species) %>%
  summarize(
    mean.height = mean(height, na.rm = T),
    mean.mass = mean(mass, na.rm = T),
    bmi.mean = mean.mass / (mean.height / 100) ^ 2
  )
sw
```

```
# A tibble: 38 x 4
   species   mean.height mean.mass bmi.mean
   <chr>           <dbl>     <dbl>    <dbl>
 1 Aleena           79       15       24.0
 2 Besalisk        198      102       26.0
 3 Cerean          198       82       20.9
 4 Chagrian        196      NaN      NaN
 5 Clawdite        168       55       19.5
 6 Droid           140       69.8     35.6
 7 Dug             112       40       31.9
 8 Ewok             88       20       25.8
 9 Geonosian       183       80       23.9
10 Gungan          209.      74       17.0
# ... with 28 more rows
```

```
sw <- starwars %>%
  group_by(species, gender) %>%
  summarize(
    mean.height = mean(height, na.rm = T),
    mean.mass = mean(mass, na.rm = T),
    bmi.mean = mean.mass / (mean.height / 100) ^ 2
  )
sw
```

```
# A tibble: 43 x 5
# Groups:   species [38]
   species   gender mean.height mean.mass bmi.mean
   <chr>     <chr>        <dbl>     <dbl>    <dbl>
 1 Aleena    male            79      15       24.0
 2 Besalisk  male           198     102       26.0
 3 Cerean    male           198      82       20.9
 4 Chagrian  male           196     NaN      NaN
 5 Clawdite  female         168      55       19.5
 6 Droid     none           200     140       35
 7 Droid     <NA>           120      46.3     32.2
```

```
 8 Dug       male           112      40       31.9
 9 Ewok      male            88      20       25.8
10 Geonosian male           183      80       23.9
# ... with 33 more rows
```

```
# same summaries, but with mutate on grouped tibble
sw <- starwars %>%
  group_by(species, gender) %>%
  mutate(
    mean.height = mean(height, na.rm = T),
    mean.mass = mean(mass, na.rm = T),
    bmi.mean = mean.mass / (mean.height / 100) ^ 2
  )
sw
```

```
# A tibble: 87 x 16
# Groups:   species, gender [43]
   name  height  mass hair_color skin_color eye_color birth_year gender
   <chr>  <int> <dbl> <chr>      <chr>      <chr>          <dbl> <chr>
 1 Luke~    172    77 blond      fair       blue            19   male
 2 C-3PO    167    75 <NA>       gold       yellow         112   <NA>
 3 R2-D2     96    32 <NA>       white, bl~ red             33   <NA>
 4 Dart~    202   136 none       white      yellow          41.9 male
 5 Leia~    150    49 brown      light      brown           19   female
 6 Owen~    178   120 brown, gr~ light      blue            52   male
 7 Beru~    165    75 brown      light      blue            47   female
 8 R5-D4     97    32 <NA>       white, red red             NA   <NA>
 9 Bigg~    183    84 black      light      brown           24   male
10 Obi-~    182    77 auburn, w~ fair       blue-gray       57   male
# ... with 77 more rows, and 8 more variables: homeworld <chr>, species <chr>,
#   films <list>, vehicles <list>, starships <list>, mean.height <dbl>,
#   mean.mass <dbl>, bmi.mean <dbl>
```

```
# same summaries, but with mutate on grouped tibble
sw <- starwars %>%
  group_by(species, gender) %>%
  mutate(
    mean.height = mean(height, na.rm = T),
    mean.mass = mean(mass, na.rm = T),
    bmi.mean = mean.mass / (mean.height / 100) ^ 2,
    bmi = mass / (height / 100) ^ 2
  )
sw
```

```
# A tibble: 87 x 17
# Groups:   species, gender [43]
   name  height  mass hair_color skin_color eye_color birth_year gender
   <chr>  <int> <dbl> <chr>      <chr>      <chr>          <dbl> <chr>
 1 Luke~    172    77 blond      fair       blue            19   male
 2 C-3PO    167    75 <NA>       gold       yellow         112   <NA>
 3 R2-D2     96    32 <NA>       white, bl~ red             33   <NA>
 4 Dart~    202   136 none       white      yellow          41.9 male
 5 Leia~    150    49 brown      light      brown           19   female
 6 Owen~    178   120 brown, gr~ light      blue            52   male
 7 Beru~    165    75 brown      light      blue            47   female
 8 R5-D4     97    32 <NA>       white, red red             NA   <NA>
```

```
 9 Bigg~    183    84 black      light      brown           24   male
10 Obi-~    182    77 auburn, w~ fair       blue-gray       57   male
# ... with 77 more rows, and 9 more variables: homeworld <chr>, species <chr>,
#   films <list>, vehicles <list>, starships <list>, mean.height <dbl>,
#   mean.mass <dbl>, bmi.mean <dbl>, bmi <dbl>
```

```
# count number of rows in group
num.sp.gend <- starwars %>%
  group_by(species, gender) %>%
  summarize(num = n())

# fraction of mass of each character
fr.mass <- starwars %>%
  group_by(species) %>%
  mutate(pct.mass = mass / sum(mass, na.rm = TRUE)) %>%
  ungroup %>%
  select(name, pct.mass)
```

Joining

```
bmi <- starwars %>%
  group_by(species) %>%
  summarize(bmi = mean(mass / (height / 100) ^ 2, na.rm = TRUE))

num.tall.characters <- starwars %>%
  filter(height > 150) %>%
  group_by(species) %>%
  summarize(num = n()) %>%
  rename(spp = "species")

num.tall.characters %>%
  left_join(bmi, by = c("spp" = "species"))
```

```
# A tibble: 31 x 3
   spp         num   bmi
   <chr>     <int> <dbl>
 1 Besalisk      1  26.0
 2 Cerean        1  20.9
 3 Chagrian      1 NaN
 4 Clawdite      1  19.5
 5 Droid         2  32.7
 6 Geonosian     1  23.9
 7 Gungan        3  16.8
 8 Human        29  25.5
 9 Hutt          1 443.
10 Iktotchi      1 NaN
# ... with 21 more rows
```

```
final <- starwars %>%
  group_by(species) %>%
  summarize(bmi = mean(mass / (height / 100) ^ 2, na.rm = TRUE)) %>%
  left_join(
    starwars %>%
      filter(height > 150) %>%
      group_by(species) %>%
      summarize(num = n()) %>%
```

```
      rename(spp = "species"),
    by = c("spp" = "species")
  )
```

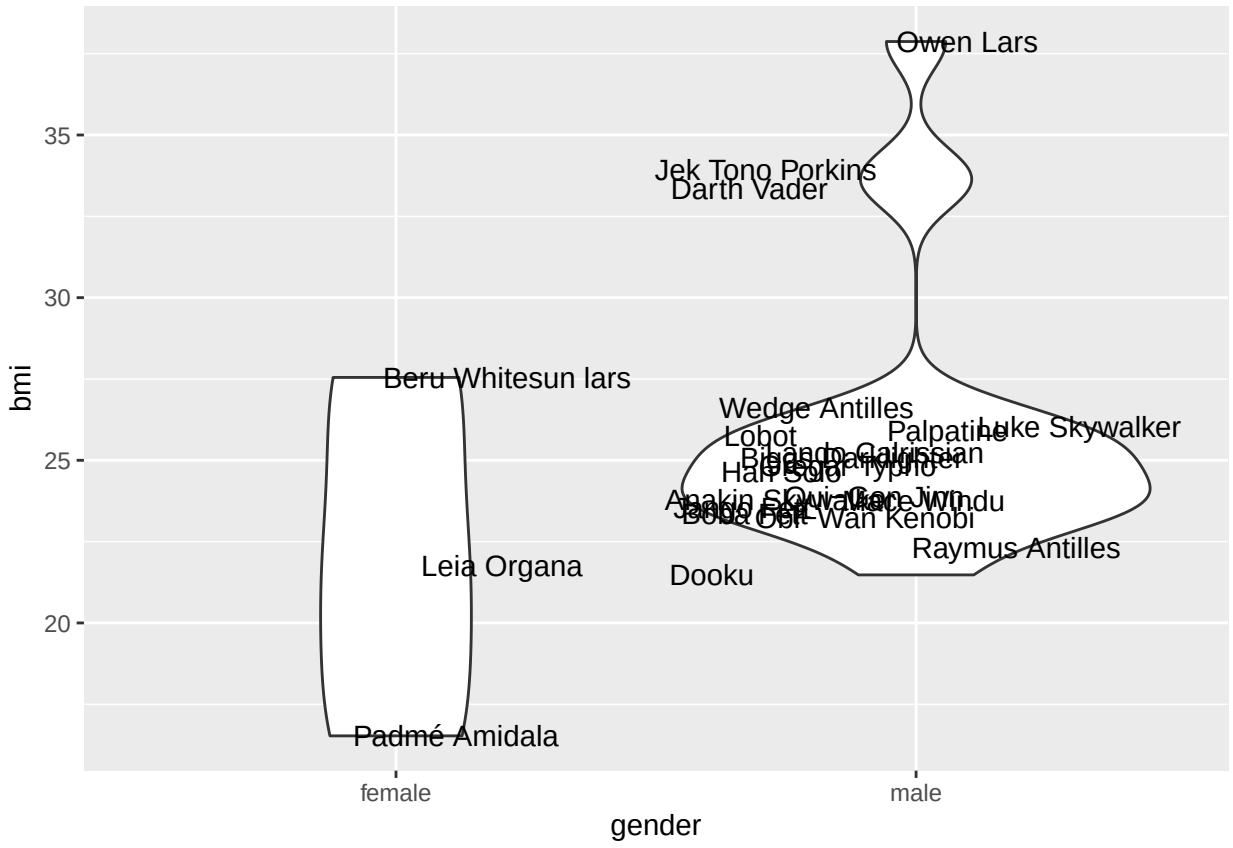
```
Error: `by` can't contain join column `spp` which is missing from LHS
```

tidyr : gather, spread

```
sw <- select(starwars, -(films:starships))

body.colors <- starwars %>%
  select(name, contains("color"))

colors.gathered <- body.colors %>%
  gather(color_type, color, -name) %>%
  arrange(name, color_type, color)

colors.spread <- colors.gathered %>%
  spread(color_type, color) %>%
  as.data.frame
```

pipeline to ggplot

```
starwars %>%
  mutate(bmi = mass / (height / 100) ^ 2) %>%
  select(name, bmi, species, gender) %>%
  filter(complete.cases(.) & species == "Human") %>%
  ggplot(aes(gender, bmi)) +
  geom_violin() +
  geom_text(aes(label = name), position = "jitter")
```


Owen Lars
Jek Tono Porkins
Darth Vader
Beru Whitesun lars
Wedge Antilles
Lobot
Palpatine
Luke Skywalker
Han Solo
Obi-Wan Kenobi
Raymus Antilles
Leia Organa
Dooku
Padmé Amidala
bmi
35
30
25
20
female
male
gender