Intro to Statistics Functions and Distributions

Revolution Analytics

Overview

In this session we'll cover the basics of statistical analysis in R. The objectives are:

- Master application of basic stats functions
- Learn about tools for exploratory data analysis
- Learn to conduct hypothesis testing, ANOVA, linear regression, and generalized linear regression in R
- Learn about sampling distributions available in R

Outline

Descriptive stats operate on vectors

```
prod(x)
sum(x)
length(x)
mean(x)
var(x)
max(x)
min(x)
range(x)
sd(x)
sort(x)
order(x)
```

Dealing with missing values

These functions are capable of handling missing values.

- See the help documentation for details.
- Specifically, see the `na.rm` argument in documentation

Outline

1. Basic statistical summaries
2. Exploratory data analysis
3. Hypothesis testing
4. Regression
5. Probability distributions
6. Other topics

Getting a quick statistical summary

You can view by-quartile summaries of vectors or elements of a data frame with `summary()`.

```
summary(mtcars$mpg)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    10.4    15.4    19.2    20.1    22.8    33.9
```

```
summary(mtcars)
```

```
##       mpg             cyl             disp             hp
##  Min.   :10.4   Min.   :4.00   Min.   : 71.1   Min.   : 52.0
##  1st Qu.:15.4   1st Qu.:4.00   1st Qu.:120.8   1st Qu.: 96.5
##  Median :19.2   Median :6.00   Median :196.3   Median :123.0
##  Mean   :20.1   Mean   :6.19   Mean   :230.7   Mean   :146.7
##  3rd Qu.:22.8   3rd Qu.:8.00   3rd Qu.:326.0   3rd Qu.:180.0
##  Max.   :33.9   Max.   :8.00   Max.   :472.0   Max.   :335.0
...
```

Correlation

You can analyze the correlation between two variables, or across a matrix of different variables, using `cor()`.

```
cor(mtcars[, 1:5], method = "pearson")
```

```
##          mpg     cyl    disp      hp    drat
## mpg   1.0000 -0.8522 -0.8476 -0.7762  0.6812
## cyl  -0.8522  1.0000  0.9020  0.8324 -0.6999
## disp -0.8476  0.9020  1.0000  0.7909 -0.7102
## hp   -0.7762  0.8324  0.7909  1.0000 -0.4488
## drat  0.6812 -0.6999 -0.7102 -0.4488  1.0000
```

```
cor(mtcars[, 1:5], method = "spearman")
```

```
##          mpg     cyl    disp      hp    drat
## mpg   1.0000 -0.9108 -0.9089 -0.8947  0.6515
## cyl  -0.9108  1.0000  0.9277  0.9018 -0.6789
## disp -0.9089  0.9277  1.0000  0.8510 -0.6836
## hp   -0.8947  0.9018  0.8510  1.0000 -0.5201
## drat  0.6515 -0.6789 -0.6836 -0.5201  1.0000
```

Exercise 1: Explore a data set

Your turn:

- Generate pairwise plots and correlation matrices for appropriate variables in `performance`
- Do you see evidence of any outliers? How should you handle them?

Hint: Experiment with `quantile()`, `hist()`, `summary()`, and `sd()`

Exercise 1 Hint: Reading the data

Use our reformatted code from earlier!

Outline

Testing for normality

There are all kinds of hypotheses that statisticians may be interested in testing. One of the most important tests is of a sample's normality:

```
cars$acceleration <- (cars$speed^2)/(2 * cars$dist)
shapiro.test(cars$acceleration)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  cars$acceleration
## W = 0.9829, p-value = 0.6772
```

Helpful Link for Interpretation:

http://www.dummies.com/how-to/content/how-to-test-data-normality-in-a-formal-way-in-r.html

Testing for differences in means

The function `t.test()` allows us to test hypotheses about differences in means

```
x.data <- rnorm(100, 3, 1)
y.data <- rnorm(100, 3, 1)
```

One-sample t-test

```
t.test(x.data)
```

```
##
##  One Sample t-test
##
## data:  x.data
## t = 34.61, df = 99, p-value < 2.2e-16
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
...
```

Helpful Link for Interpretation:

`http://www.stat.columbia.edu/~martin/W2024/R2.pdf`

Use `help(t.test)` to find additional arguments (esp. see `paired`)

Independent samples t-test

```
t.test(x.data, y.data)
```

```
##
##  Welch Two Sample t-test
##
## data:  x.data and y.data
## t = 1.117, df = 197.2, p-value = 0.2653
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -0.1123  0.4057
## sample estimates:
## mean of x mean of y
##     3.109     2.962
```

Helpful Link for Interpretation:

http://www.stat.columbia.edu/~martin/W2024/R2.pdf

Exercise: Two-sample t-test

```
library(ISwR)
```

The package `ISwR` gives you access to the dataset intake, pre- and post-menstrual caloric intake of a group of women.

- Calculate the group intake means, then use `t.test()` to calculate whether the difference between groups is statistically significant.
- Is the post-pre difference normally distributed?

Outline

Using `lm()` for regression

Let's use `lm()` to model `mpg` in the `mtcars` dataset.

```
fit0 <- lm(mpg ~ ., data = mtcars)  # fit with all variables
fit0
```

```
##
## Call:
## lm(formula = mpg ~ ., data = mtcars)
##
## Coefficients:
## (Intercept)          cyl         disp           hp         drat
##     12.3034      -0.1114       0.0133      -0.0215       0.7871
##          wt         qsec           vs           am         gear
##     -3.7153       0.8210       0.3178       2.5202       0.6554
##        carb
##     -0.1994
```

How to extract inference results?

```
summary(fit0)
```

```
##
## Call:
## lm(formula = mpg ~ ., data = mtcars)
##
## Residuals:
##    Min     1Q Median     3Q    Max
##  -3.45  -1.60  -0.12   1.22   4.63
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  12.3034    18.7179    0.66    0.518
## cyl          -0.1114     1.0450   -0.11    0.916
## disp          0.0133     0.0179    0.75    0.463
## hp           -0.0215     0.0218   -0.99    0.335
## drat          0.7871     1.6354    0.48    0.635
## wt           -3.7153     1.8944   -1.96    0.063 .
## qsec          0.8210     0.7308    1.12    0.274
## vs            0.3178     2.1045    0.15    0.881
## am            2.5202     2.0567    1.23    0.234
## gear          0.6554     1.4933    0.44    0.665
## carb         -0.1994     0.8288   -0.24    0.812
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.65 on 21 degrees of freedom
## Multiple R-squared:  0.869,  Adjusted R-squared:  0.807
## F-statistic: 13.9 on 10 and 21 DF,  p-value: 0.000000379
```

Uses a `formula`

A concept that is relevant in many different aspects of R

- Two sides separated by a "~" symbol
- Left side: dependent variables (in this example, it is `mpg`)
- Right side: independent variables, separated by a "+" in simple cases (in this example a `.` means all other variables in the dataset).

Examples:

- mpg ~ hp
- mpg ~ hp + disp
- hp ~ cyl + am + wt

Exercise 2: formulae

- Create a formula that specifies a model that predicts `mpg` by `vs`, `gear`, and `wt`
- Use `lm()` as above to estimate this model, and extract some inference results.

Using factor variables in regression

Several of the numeric variables in `mtcars`, e.g. `cyl`, `gear`, and `carb` are better expressed as factors.

```
new.data <- mtcars
factor.cols <- c("cyl", "gear", "carb")
new.data[, factor.cols] <- lapply(new.data[, factor.cols], as.factor)
```

re-fit the model with factor columns

Several of the numeric variables in `mtcars`, e.g. `cyl`, `gear`, and `carb` are better expressed as factors.

```
fit1 <- lm(mpg ~ ., data = new.data)
summary(fit1)
```

```
##
## Call:
## lm(formula = mpg ~ ., data = new.data)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -3.509 -1.358 -0.095  0.775  4.625
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  23.8791    20.0658    1.19    0.253
## cyl6         -2.6487     3.0409   -0.87    0.397
## cyl8         -0.3362     7.1595   -0.05    0.963
## disp          0.0355     0.0319    1.11    0.283
## hp           -0.0705     0.0394   -1.79    0.094 .
## drat          1.1828     2.4835    0.48    0.641
## wt           -4.5298     2.5387   -1.78    0.095 .
## qsec          0.3678     0.9354    0.39    0.700
## vs            1.9309     2.8713    0.67    0.512
...
```

Simplify Model with stepwise regression

With the MASS package, we can use the function stepAIC() To do model selection.

```
library(MASS)  # package associated with Modern Applied Statistics with S
stepAIC(fit1, direction = "both")
```

```
## Start:  AIC=76.4
## mpg ~ cyl + disp + hp + drat + wt + qsec + vs + am + gear + carb
##
##         Df Sum of Sq RSS  AIC
## - carb  5     13.60 134 69.8
## - gear  2      3.97 124 73.4
## - am    1      1.14 122 74.7
...
```

```
##
## Call:
## lm(formula = mpg ~ cyl + hp + wt + am, data = new.data)
##
## Coefficients:
## (Intercept)         cyl6         cyl8           hp           wt
##     33.7083      -3.0313      -2.1637      -0.0321      -2.4968
...
```

Simplify Model with stepwise regression

Now explicitly fit the best model

```
fit2 <- lm(mpg ~ cyl + hp + wt + am, data = new.data)
summary(fit2)
```

```
##
## Call:
## lm(formula = mpg ~ cyl + hp + wt + am, data = new.data)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -3.939 -1.256 -0.401  1.125  5.051
##
## Coefficients:
##             Estimate Std. Error t value         Pr(>|t|)
## (Intercept)  33.7083     2.6049   12.94 0.00000000000077 ***
## cyl6         -3.0313     1.4073   -2.15           0.0407 *
## cyl8         -2.1637     2.2843   -0.95           0.3523
## hp           -0.0321     0.0137   -2.35           0.0269 *
## wt           -2.4968     0.8856   -2.82           0.0091 **
## am            1.8092     1.3963    1.30           0.2065
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
...
```

Getting regression summaries: `fit1`

```r
summary(fit1)$adj.r.squared
```

```
## [1] 0.779
```

```r
summary(fit1)$coefficients
```

```
##             Estimate Std. Error  t value Pr(>|t|)
## (Intercept) 23.87913   20.06582  1.19004  0.25253
## cyl6        -2.64870    3.04089 -0.87103  0.39747
## cyl8        -0.33616    7.15954 -0.04695  0.96317
## disp         0.03555    0.03190  1.11433  0.28267
## hp          -0.07051    0.03943 -1.78835  0.09393
## drat         1.18283    2.48348  0.47628  0.64074
## wt          -4.52978    2.53875 -1.78426  0.09462
## qsec         0.36784    0.93540  0.39325  0.69967
## vs           1.93085    2.87126  0.67248  0.51151
## am           1.21212    3.21355  0.37719  0.71132
## gear4        1.11435    3.79952  0.29329  0.77332
## gear5        2.52840    3.73636  0.67670  0.50890
## carb2       -0.97935    2.31797 -0.42250  0.67865
## carb3        2.99964    4.29355  0.69864  0.49547
## carb4        1.09142    4.44962  0.24528  0.80956
## carb6        4.47757    6.38406  0.70137  0.49381
## carb8        7.25041    8.36057  0.86722  0.39948
```

Getting regression summaries: `fit2`

```r
summary(fit2)$adj.r.squared
```

```
## [1] 0.8401
```

```r
summary(fit2)$coefficients
```

```
##             Estimate Std. Error t value          Pr(>|t|)
## (Intercept) 33.70832    2.60489 12.9404 0.0000000000007733
## cyl6        -3.03134    1.40728 -2.1540 0.0406827179363793
## cyl8        -2.16368    2.28425 -0.9472 0.3522508691483529
## hp          -0.03211    0.01369 -2.3450 0.0269346052360614
## wt          -2.49683    0.88559 -2.8194 0.0090814075583834
## am           1.80921    1.39630  1.2957 0.2064596737699271
```

Anova

The function `anova()` allows us to test whether the full model is explains significantly more variance than the pruned one.

```
anova(fit2, fit1)
```

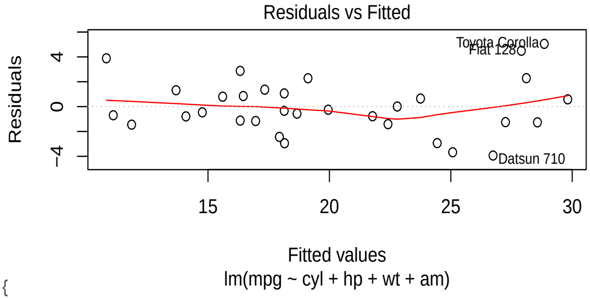
```
## Analysis of Variance Table
##
## Model 1: mpg ~ cyl + hp + wt + am
## Model 2: mpg ~ cyl + disp + hp + drat + wt + qsec + vs + am + gear + carb
##   Res.Df RSS Df Sum of Sq    F Pr(>F)
## 1     26 151
## 2     15 120 11      30.6 0.35   0.96
```

REVOLUTION ANALYTICS

Plotting regression results

and `plot()` has a method for residual analysis of linear models:

```
plot(fit2, 1)
```



{

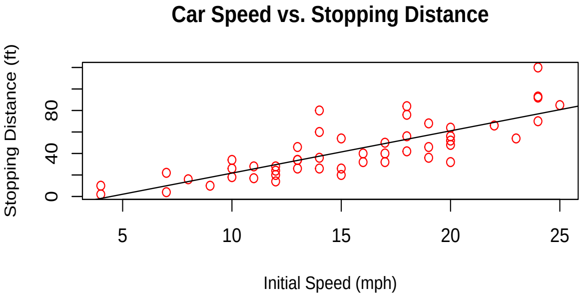
}

Adding regression lines

As an aside, regression lines can easily be added to your R plots.

Recall last module's `cars` example:

```
plot(dist ~ speed, data = cars, xlab = "Initial Speed (mph)", ylab = "Stopping Distance (ft)",
  main = "Car Speed vs. Stopping Distance", col = "red", cex.lab = 0.9)
regression.line <- lm(dist ~ speed, data = cars)
abline(regression.line)
```



{

}

Exercise

What variables best explain ROI? Your turn, using the `performance` data.

- What factors in the `performance` data set are the strongest predictors of `ROI`?
  - Hint: Explore the data, transform the data, then model it using `lm()`.
- Can you simplify your original model?
  - Hint: Experiment with `stepAIC()` to refine your model.
- Bonus points: Can you efficiently apply this analysis to the three subsets of performance?

http://www.stat.cmu.edu/~cshalizi/402/programming/writing-functions.pdf

Outline

Probability distributions

R supports all of the well known distribution functions

- Uniform
- Beta
- Binomial
- Chi-squared
- Exponential
- Poisson
- F-distribution
- Gamma
- Geometric
- Hypergeometric
- Log-normal

Normal distribution function example

`dnorm()` provides the probability density function (PDF)

```
dnorm(5, mean = 0, sd = 1)  # height of PDF at x = 5
```

```
## [1] 0.000001487
```

`pnorm()` provides the distribution function (CDF)

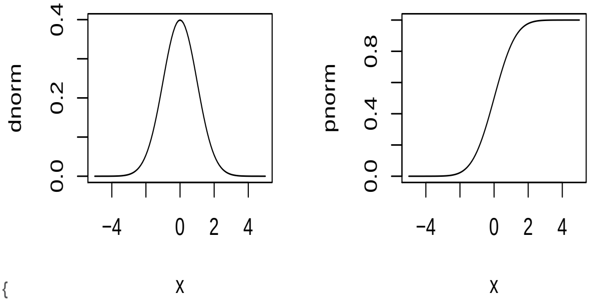
```
pnorm(5, mean = 0, sd = 1)  # P(X<5 | X~N(0,1) )
```

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

Plotting distributions

The function `plot()` also has a method for plotting distributions, e.g.:

```
par(mfrow = c(1, 2))
plot(dnorm, -5, 5)
plot(pnorm, -5, 5)
```




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

Sampling from a distribution

`rnorm()` sample from a normal distribution with a particular `mean` and `sd`

```
rnorm(10, mean = 0, sd = 1)
```

```
##  [1]  0.4094  1.6889  1.5866 -0.3309 -2.2852  2.4977  0.6671  0.5413
##  [9] -0.0134  0.5101
```

Reproducible results

To make the results reproducible we can use the seed value:

```
set.seed(100)
rnorm(5)
```

```
## [1] -0.50219  0.13153 -0.07892  0.88678  0.11697
```

```
rnorm(5)
```

```
## [1]  0.3186 -0.5818  0.7145 -0.8253 -0.3599
```

```
set.seed(100)
rnorm(5)
```

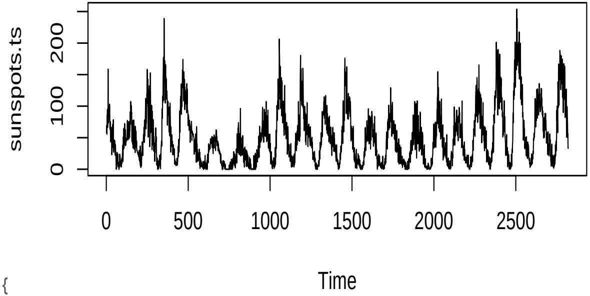
```
## [1] -0.50219  0.13153 -0.07892  0.88678  0.11697
```

Outline

Time series data

R has a native time series data type

```
sunspots.ts <- ts(sunspots)
plot(sunspots.ts)
```



{

}

Time series data

Time Series objects allow you to perform interesting analysis like spectral analysis or ARMA and ARIMA models.

We won't cover time series in this course but there are some good resources for time series

Time Series Analysis and Its Applications: With R Examples by Robert H. Shumway, David S. Stoffer

There is much, much more in R

There's pretty much nothing out there that can be done with statistics that R doesn't handle.

Other topics:

- logistic regression
- machine learning
- cluster analysis
- survival analysis

See other Revolution course material for in-depth tutorials on these topics and more.

Module review questions

- What are some of the basic functions available to users for exploratory data analysis?
- What are some ways to visually and empirically test a sample’s distribution?
- What are some examples of useful plot methods beyond simple scatterplots?
- How do you exactly reproduce the results of a random experiment in R?

Thank you

Revolution Analytics is the leading commercial provider of software and support for the popular open source R statistics language.

www.revolutionanalytics.com
1.855.GET.REVO
Twitter: @RevolutionR