

Getting Started with the New Statistics in R

David Erceg-Hurn, Geoff Cumming, Robert Calin-Jageman

2016-08-11

Overview

R is a popular and powerful free program that can be used to conduct most of the statistical analyses outlined in *Introduction to the New Statistics* (Cumming & Calin-Jageman, 2017). Unlike programs such as SPSS where analyses are usually conducted by clicking on menus, in R analyses are typically performed by typing *commands*.

This document is a brief guide that will help you to get started using the 'new statistics' in R. The guide is split into three sections. The first part provides some tips about installing and learning the basics of R. If you've never used R before you should read this section - if you already know how to use R you can skip it. The second section provides a brief overview of a new R package, *itns*, that contains the datasets used in *Introduction to the New Statistics*. You can use the datasets in the *itns* package to work through the examples covered in the book and the end-of-chapter exercises. The final section provides an overview of R packages and functions that can be used to conduct the analyses covered in *Introduction to the New Statistics*.

You will notice that some words in this document are a blue colour. These are hyper links. If you click on the blue text, you will be redirected to websites that contain information about using R.

Part One - Installing R and Learning the Basics

To install R, visit the [RStudio website and follow the installation instructions](#). That webpage also contains links to interactive tutorials for R beginners. The tutorials will help you learn how to perform basic tasks like importing and manipulating datasets. Other useful resources for learning R include:

- [R for Data Science](#) - An online book by Garrett Grolemund and Hadley Wickham that will teach you how to import, tidy, and explore data.
- [Kelly Black's R Tutorial](#) - An introductory tutorial focusing on the basics of R.
- [How to Learn R Blog](#) - A collection of resources that will help you learn R.
- [Quick-R](#) - A website that contains example code for running basic analyses.
- [R Quick Reference Card](#) - A list of key commands built into R.

Also remember that Google is your friend. If you have a question about how to do something in R, it is likely that someone else has already asked the same question and that there is an answer on the Internet. For example if you type 'R how to create a histogram' into Google, you will find many links to webpages showing you the R code that you need to plot a histogram.

In the remainder of the document, we assume that you have a basic understanding of how to use R.

Part Two - The *itns* Package

itns is an R package that contains most of the datasets used in *Introduction to the New Statistics*. The datasets were converted from Microsoft Excel files (found on the book's website) into R data frames. The table on the next page lists the names of the data frames in the package, and the sections of the book where they are mentioned.

itns Data Frames For Within-Chapter Exercises

Name	Chapter	Topic
body_well	11, 12	Correlation, Regression
natsal	16.11	Robust Methods - Two Independent Groups
pen_laptop1	7.6-7.12	Two Independent Groups
pen_laptop2	7.36-7.38	Two Independent Groups
rattan	14.10-14.12	One-Way Independent Group Contrasts and Comparisons
thomason1	8, 11, 12	Two Dependent Groups, Scatterplots, Regression
thomason2	8	Two Dependent Groups
thomason3	8, 12.18	Two Dependent Groups, Regression

itns Data Frames For End-Of-Chapter Exercises

Name	Chapter	Topic
altruism_happiness	12.3	Regression
anchor_estimate	7.3	Two Independent Groups
anchor_estimate_ma	9.1	Meta-Analysis
campus_involvement	11.7	Correlation
clean_moral_johnson	7.4	Two Independent Groups
clean_moral_schall	7.4 , 10.2	Two Independent Groups
college_survey1	3.2, 3.3, 5.2, 5.3	Descriptive Statistics & Plots, Single Sample Confidence Interval
college_survey2	5.4	Single Sample Confidence Interval
dana	16.3	Robust Methods - Two Independent Groups
emotion_hearttrate	8.3	Two Dependent Groups
exam_scores	11.2	Correlation
flag_priming_ma	9.2	Meta-Analysis
home_prices	12.2	Regression
home_prices_holdout	12.2	Regression
labels_flavor	8.4	Two Dependent Groups
math_gender_iat	7.5	Two Independent Groups
math_gender_iat_ma	9.3	Meta-Analysis
organic_moral	14.7	One-Way Independent Group Contrasts and Comparisons
power_performance_ma	9.4	Meta-Analysis
religious_belief	3.4	Descriptive Statistics & Plots
religion_sharing	14.3	One-Way Independent Group Contrasts and Comparisons
self_explain_time	15.4	Analysing factorial designs
sleep_beauty	11.6	Correlation
study_strategies	14.1	One-Way Independent Group Contrasts and Comparisons
stickgold	6.5	Single Sample Confidence Interval
videogame_aggression	15.3	Analysing factorial designs

The `itns` package is not yet on [CRAN](#), but you can download it from [github](#) using the `devtools` package:

```
# install.packages("devtools") # if you haven't already, install devtools from CRAN
library(devtools)              # load devtools
install_github("gitrman/itns") # install itns
```

Once you have installed the package, you can use the `library()` function to load it, `str()` to examine metadata for each data frame, and functions such as `head()` and `tail()` to print the first or last few rows to your screen.

```
library(itns)      # loads the package
str(pen_laptop1)   # displays metadata
```

```
## 'data.frame':   65 obs. of  2 variables:
## $ group      : Factor w/ 2 levels "Laptop","Pen": 2 2 2 2 2 2 2 2 2 ...
## $ transcription: num  12.1 6.5 8.1 7.6 12.2 10.8 1 2.9 14.4 8.4 ...
```

```
head(pen_laptop1) # prints the first few rows
```

```
##   group transcription
## 1   Pen           12.1
## 2   Pen           6.5
## 3   Pen           8.1
## 4   Pen           7.6
## 5   Pen          12.2
## 6   Pen          10.8
```

```
tail(pen_laptop1) # prints the last few rows
```

```
##   group transcription
## 60 Laptop          10.3
## 61 Laptop           9.0
## 62 Laptop          12.8
## 63 Laptop          12.0
## 64 Laptop          34.7
## 65 Laptop           4.1
```

To access further details about each dataset, type a question mark and the name of the dataset, for example:

```
?pen_laptop1
```

or access the PDF help file [LINK TO GO HERE](#) on the [itns github site](#).

The datasets in the `itns` package can be used to replicate analyses that appear in *Introduction to the New Statistics*, and to work through the book's end-of-chapter exercises using the packages and functions outlined in the next section of this guide.

Part 3 - Helpful Packages and Functions

Most of the analyses described in *Introduction to the New Statistics* can be conducted using inbuilt R functions, or functions in packages that can be downloaded from CRAN or github. In this section we mention some useful functions and packages, and resources that will help you learn how to use them. This section is *not* intended to be a comprehensive tutorial on how to use each function; rather, our aim is to point you in the direction of resources already on the Internet.

Basic Descriptive Statistics

Functions to compute basic descriptive statistics are built into R. These include `mean()`, `median()`, `minimum()` and `maximum()` functions; `var()` for variance, `sd()` for the standard deviation, `IQR()` for interquartile range, `range()`, `quantile()` for percentiles, and `summary()`, which for numeric variables returns the minimum, 25th percentile, median, mean, 75th percentile, and maximum. Some examples of these functions in action are given below. See [John Quirk's tutorial](#) on using basic descriptive statistics for more information.

```

# Compute basic descriptive statistics for transcription score in pen_laptop1 data frame
# Mean
mean(pen_laptop1$transcription)

## [1] 11.53385

# Median
median(pen_laptop1$transcription)

## [1] 10.7

# Standard Deviation
sd(pen_laptop1$transcription)

## [1] 6.690695

# 0 to 100th percentile in steps of 10%
quantile(pen_laptop1$transcription, probs = seq(from = 0, to = 1, by = .1))

##      0%      10%      20%      30%      40%      50%      60%      70%      80%      90%     100%
##  1.00   3.38   6.24   8.50   9.16  10.70  12.04  13.20  17.06  18.92  34.70

# Example of summary function output
summary(pen_laptop1$transcription)

##      Min. 1st Qu.  Median     Mean 3rd Qu.     Max.
##      1.00   8.00   10.70   11.53   15.20   34.70

```

Summary Statistics By Group

You will sometimes want to compute descriptive statistics separately for multiple groups. There are many ways to do this. One option is to use the `group_by()` and `summarise()` functions in the `dplyr` package, for example:

```

# Compute mean and standard deviation separately for the laptop and pen groups
library(dplyr)

##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##      filter, lag

## The following objects are masked from 'package:base':
##
##      intersect, setdiff, setequal, union

```

```
pen_laptop1 %>%
  group_by(group) %>%
  summarise(
    mean = mean(transcription),
    sd = sd(transcription)
  )
```

```
## Source: local data frame [2 x 3]
##
##   group      mean      sd
##   (fctr)    (dbl)    (dbl)
## 1 Laptop 14.519355 7.285576
## 2   Pen   8.811765 4.749339
```

For more information see the section on *Grouped Operations* in the [dplyr tutorial](#).

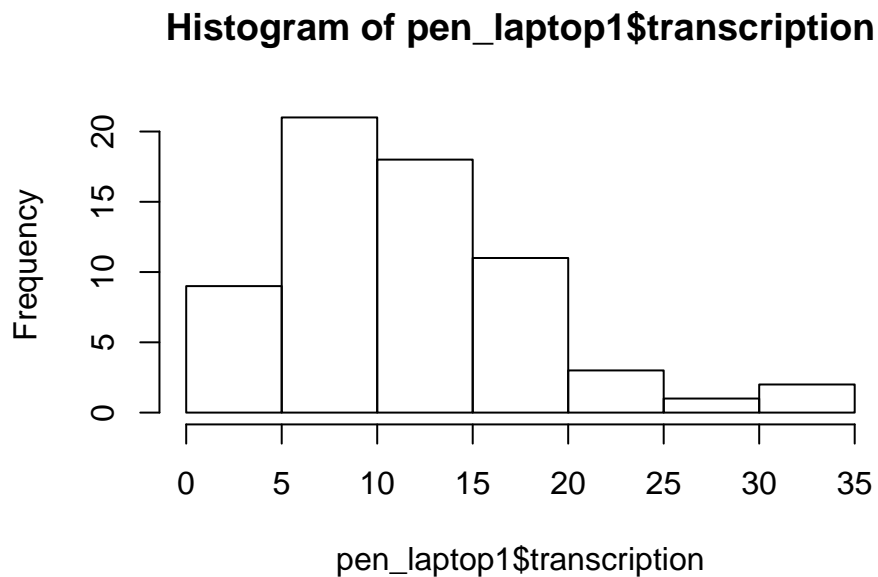
Other options for computing descriptive statistics separately for different groups include the inbuilt R function [aggregate\(\)](#) or the [doBy](#) package.

Data Visualisation (Plotting)

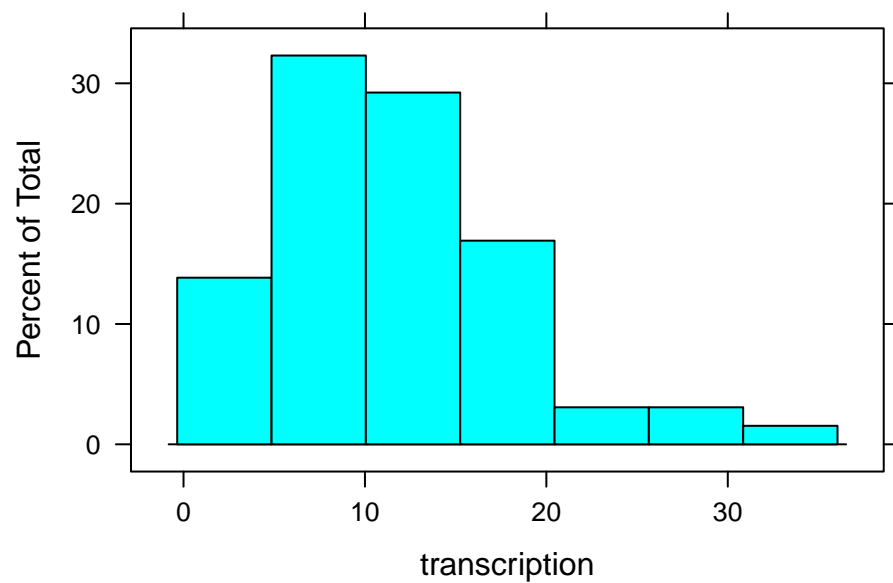
R has three systems that can be used for data visualisation - [Base graphics](#), [lattice](#), and [ggplot2](#). The STHDA website has [guides to creating graphics](#) using all three systems.

Base graphics, lattice, and ggplot2 all have functions for creating histograms and dotplots, covered in Chapter 3 of *Introduction to the New Statistics*. Here are some examples of simple histograms produced by the three packages:

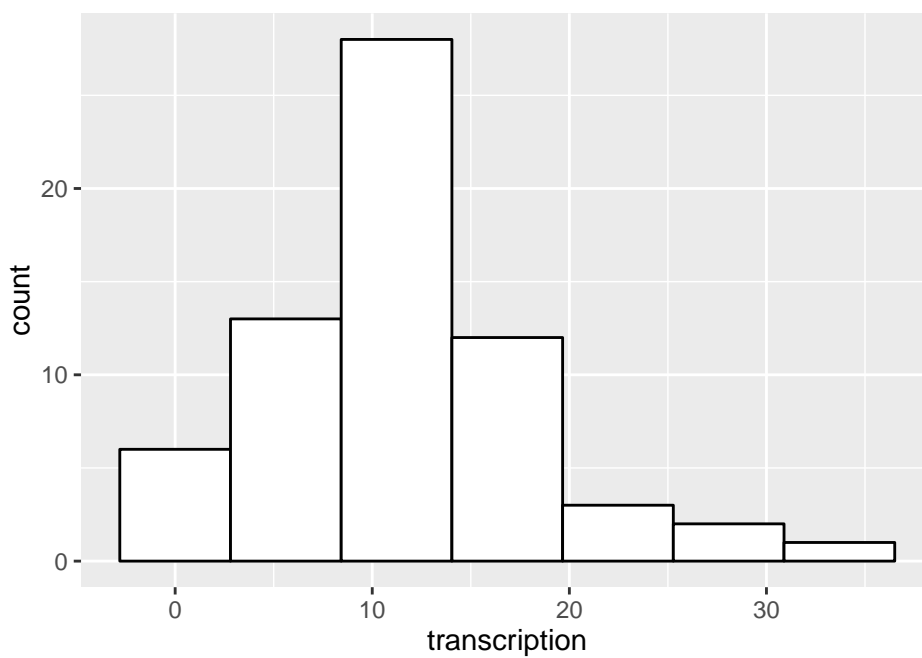
```
# Base Graphics Histogram
hist(pen_laptop1$transcription)
```



```
# lattice Histogram
library(lattice)
histogram(~transcription, data = pen_laptop1)
```



```
# ggplot2 Histogram
library(ggplot2)
ggplot(pen_laptop1, aes(transcription)) + geom_histogram(bins = 7, colour="black", fill="white")
```



If you are new to R and want to learn one graphics package, we recommend learning how to use `ggplot2` as it is the most powerful and flexible system. Resources that will help you learn how to use `ggplot2` include:

- [Winston Chang's R Graphics Cookbook](#).
- [STHDA's ggplot2 essentials](#).
- Hadley Wickham's [ggplot2 book](#).
- [DataCamp's ggplot2 courses](#).
- [Harvard Introduction to ggplot2](#).
- [R4Stats ggplot2 tutorial](#).
- [ggplot2 online documentation](#).
- [R-Studio's Data Visualisation cheatsheet](#).
- An [online workshop](#) about creating publication quality graphics using the `ggplot2` and `lattice` graphics packages by Tim Appelhans.

If you are interested in learning the `lattice` package, a good place to start is the [STHDA Lattice Guide](#). R-Studio also have a handy [Guide to R Graphics using lattice](#). There is also a [book about the Lattice package](#).

ggplot2 Histogram and Dotplot Tutorials

- [R Bloggers - How to make a histogram with ggplot2](#).
- [STHDA Histogram Tutorial](#).
- [STHDA guide to making dotplots](#).
- [ggplot2 documentation](#) for the `geom_dotplot()` geom.

Cat's Eye Pictures and Difference Plots

The [multicon](#) package, available on CRAN, contains the functions `egraph()`, `catseye()` and `diffPlot()`. These can be used to quickly produce plots of error bars, cat's eye pictures, and difference plots.

```
#install.packages("multicon") # if needed , install multicon package  
library(multicon) # load the package
```

```
## Loading required package: psych
```

```
##
```

```
## Attaching package: 'psych'
```

```
## The following objects are masked from 'package:ggplot2':
```

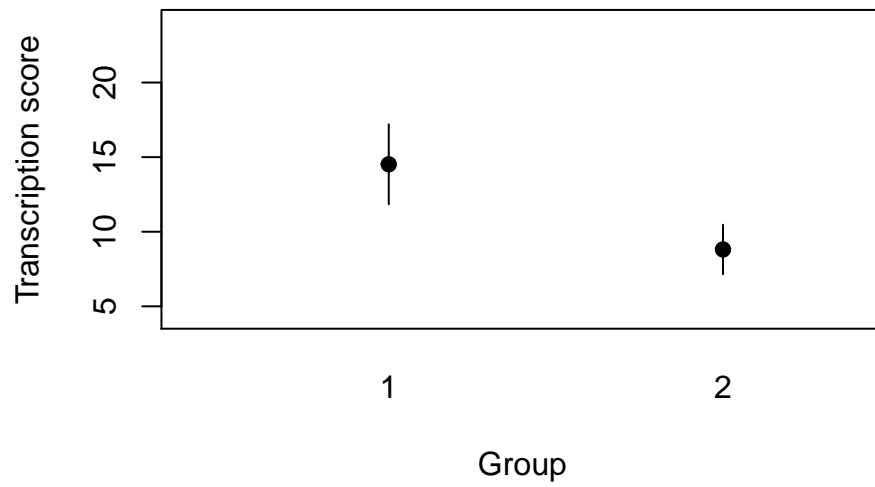
```
##
```

```
##      %+%, alpha
```

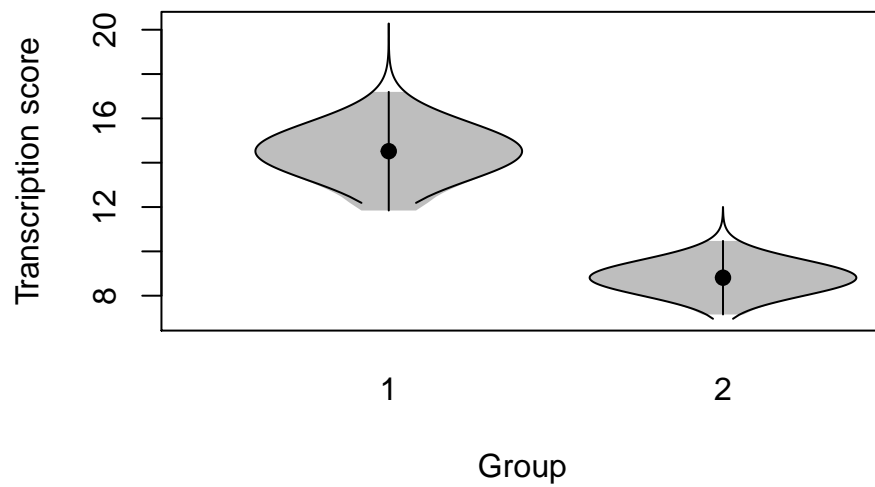
```
## Loading required package: abind
```

```
## Loading required package: foreach
```

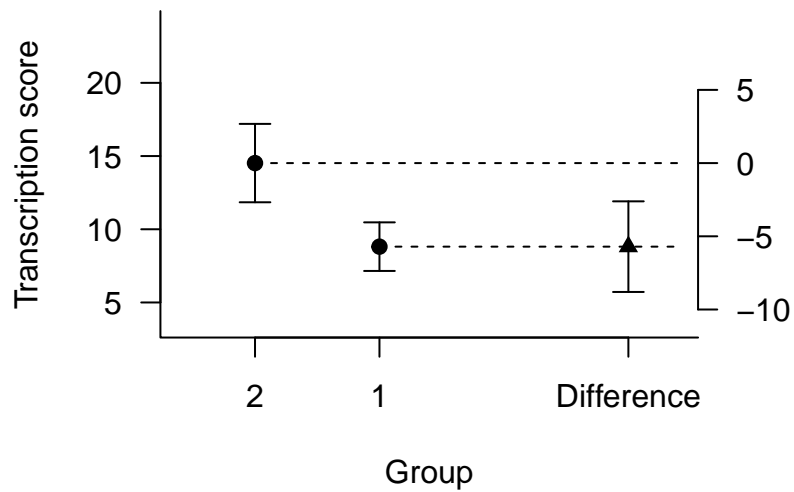
```
# Plot group means and 95% confidence intervals  
egraph(DV = pen_laptop1$transcription, grp = pen_laptop1$group,  
       xlab = "Group", ylab = "Transcription score")
```



```
# Create a Cat's Eye Plot
catseye(DV = pen_laptop1$transcription, grp = pen_laptop1$group,
        xlab = "Group", ylab = "Transcription score")
```



```
# Create a Difference Plot
diffPlot(transcription ~ group, data = pen_laptop1,
        xlab = "Group", ylab = "Transcription score")
```

Z-Scores

[John Quick's tutorial](#) shows how to use R's inbuilt `scale()` function to compute Z-scores. See also [Seam Dolinar's tutorial](#) on calculating Z scores and finding tail probabilities.

P-values and Confidence Intervals for a Single Sample

Kelly Black has written tutorials showing how to [compute p values using z- or t-distributions](#), and how to [calculate confidence intervals for means using normal or t-distributions](#).

t.test() function

The `t.test()` function is built into R. It produces confidence intervals and p-values for single samples, two independent groups, and paired samples.

```
# Single Sample
t.test(pen_laptop1$transcription)

##
## One Sample t-test
##
## data: pen_laptop1$transcription
## t = 13.898, df = 64, p-value < 2.2e-16
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##  9.875973 13.191719
## sample estimates:
## mean of x
## 11.53385
```

```
# Two Independent Groups - by default the Welch T-Test (equal variances not assumed) is calculated  
t.test(transcription ~ group, data = pen_laptop1)
```

```
##  
## Welch Two Sample t-test  
##  
## data: transcription by group  
## t = 3.7031, df = 50.816, p-value = 0.0005254  
## alternative hypothesis: true difference in means is not equal to 0  
## 95 percent confidence interval:  
## 2.612991 8.802189  
## sample estimates:  
## mean in group Laptop mean in group Pen  
## 14.519355 8.811765
```

```
# Two Independent Groups - assuming variances are equal  
t.test(transcription ~ group, data = pen_laptop1, var.equal = TRUE)
```

```
##  
## Two Sample t-test  
##  
## data: transcription by group  
## t = 3.7738, df = 63, p-value = 0.0003579  
## alternative hypothesis: true difference in means is not equal to 0  
## 95 percent confidence interval:  
## 2.685265 8.729915  
## sample estimates:  
## mean in group Laptop mean in group Pen  
## 14.519355 8.811765
```

```
# Paired Samples  
t.test(thomason1$pre, thomason1$post, paired = TRUE)
```

```
##  
## Paired t-test  
##  
## data: thomason1$pre and thomason1$post  
## t = -3.8555, df = 11, p-value = 0.002674  
## alternative hypothesis: true difference in means is not equal to 0  
## 95 percent confidence interval:  
## -2.618115 -0.715218  
## sample estimates:  
## mean of the differences  
## -1.666667
```

MBESS package

Ken Kelley's [MBESS \(Methods for the Behavioural and Social Sciences\)](#) package contains numerous functions which can be used to compute confidence intervals for many effect sizes, including standardized mean differences, mean contrasts in one-way and factorial designs, unstandardized and standardised regression coefficients, R-squared, etc. MBESS also includes functions for power analysis and sample size planning for precision. The [MBESS website](#) contains links to two journal articles about the package, and help files.

effsize package

The MBESS package contains the functions `smd()` and `ci.smd()`, which can be used to compute the standardized mean difference for the two independent groups design, and a confidence interval. However, using MBESS for this task is somewhat cumbersome as the point estimate and confidence interval have to be calculated in separate steps. The sample size for each group must also be calculated.

```
# Use dplyr package to extract transcription scores for
# the laptop and pen groups in the pen_laptop1 dataset
# library(dplyr) # load dplyr if it has not already been loaded
laptop <- pen_laptop1 %>% filter(group == "Laptop")
pen <- pen_laptop1 %>% filter(group == "Pen")
# Install MBESS if it is not already installed
# install.packages("MBESS")
# Load MBESS library
library(MBESS)
# Use the smd() function to compute d-biased (Cohen's d)
es <- smd(laptop$transcription, pen$transcription)
# Sample sizes
n1 <- nrow(pen)
n2 <- nrow(laptop)
# Use ci.smd() to compute a 95% confidence interval for the biased estimate
ci.smd(smd = es, n.1 = n1, n.2 = n2)
```

```
## $Lower.Conf.Limit.smd
## [1] 0.4204238
##
## $smd
## [1] 0.9371681
##
## $Upper.Conf.Limit.smd
## [1] 1.447208
```

A faster way to compute the standardized mean difference and confidence interval is to use the `cohen.d()` function in the [effsize](#) package, which can be downloaded from CRAN.

```
# Install effsize package if it is not already installed
#install.packages("effsize")
# Load library
library(effsize)
# Compute d-biased
cohen.d(transcription ~ group, data = pen_laptop1, noncentral = TRUE)

## t: 3.77382 df: 63
## ncp1: 5.827667
## ncp2: 1.692981
##
## Cohen's d
##
## d estimate: 0.9371681 (large)
## 95 percent confidence interval:
## inf sup
## 0.4204249 1.4472085
```

```
# Compute d-unbiased
cohen.d(transcription ~ group, data = pen_laptop1, noncentral = TRUE, hedges.correction = TRUE)

## t: 3.77382 df: 63
## ncp1: 5.827667
## ncp2: 1.692981

##
## Hedges's g
##
## g estimate: 0.9259669 (large)
## 95 percent confidence interval:
##      inf      sup
## 0.4204249 1.4472085
```

Cohen's d for Repeated Measures Designs

The *itns* package contains a function called `cohensd_rm()` that you can use to compute Cohen's d and a confidence interval for repeated measures (paired samples).

```
# Compute Cohen's d and a 95% CI
cohensd_rm(x = thomason1$post, y = thomason1$pre)
```

```
##      est      ll      ul
## 0.5354272 0.1855364 0.9178763
```

In the function output, *est* is the estimated effect size (d-value), *ll* is the lower limit of the confidence interval, and *ul* is the upper limit of the interval.

By default, the function computes a 95% confidence interval. To use a different confidence level, use the argument 'ci'. For example, to compute a 99% confidence interval you would use the following code:

```
# Compute Cohen's d and a 99% CI
cohensd_rm(x = thomason1$post, y = thomason1$pre, ci = 99)
```

```
##      est      ll      ul
## 0.5354272 0.07550802 1.03762241
```

It is also possible to correct the estimate of d for small sample bias, using the *unbiased* argument.

```
# Compute unbiased Cohen's d and a 95% CI
cohensd_rm(x = thomason1$post, y = thomason1$pre, unbiased = TRUE)
```

```
##      est      ll      ul
## 0.4979258 0.1855364 0.9178763
```

The `cohensd_rm()` function uses the average of the pre and post-treatment scores as the standardizer. This is the standardizer recommended in *Introduction to the New Statistics*. An alternative (which we do not recommend) is to use the standard deviation of the change scores as the standardizer. Should you wish to do this, you can use the `cohen.d()` function in the *effsize* package.

```

cohen.d(thomason1$pre, thomason1$post, noncentral = TRUE, paired = TRUE)

## t: 3.855498 df: 11
## ncp1: 6.328024
## ncp2: 1.279131

##
## Cohen's d
##
## d estimate: -1.112986 (large)
## 95 percent confidence interval:
##      inf      sup
## 0.3856724 1.9079710

```

Meta-Analysis

There are numerous R packages that can be used to conduct meta-analyses for a wide variety of effect sizes such as means, mean differences, standardized mean differences, proportions, odds ratios, etc. See the [CRAN Meta-Analysis Task View](#) for a comprehensive list of them.

A popular and well documented package for conducting meta-analyses in R is [metafor](#). See the detailed [metafor website](#) for more information.

The [compute.es](#) package can be used to compute and convert between various effect sizes as part of performing a meta-analysis.

[metagear](#) is a relatively new package that has meta-analytic capabilities, as well as functions that help users conduct systematic reviews and generate [PRISMA \(Preferred Reporting Items for Systematic Reviews and Meta-Analyses\)](#) flow charts. [This vignette](#) provides an overview of the metagear package.

Other useful sources of information about conducting meta-analyses in R include:

- [A.C Del Re's Practical Tutorial](#) on conducting Meta-Analysis in R using the metafor and MAd packages
- Stephanie Kovalchik's [Tutorial on Meta-Analysis in R](#) from the 2013 useR! Conference
- Schwarzer, Carpenter, and Rucker's [Meta-Analysis with R](#) book
- [R-Studio's tutorial](#) on running meta-analyses in R using the metafor package
- Simon Knight's Guide to Meta-Analysis in R - [part 1](#) and [part 2](#).
- Stephanie Hick's [Easy Introduction to Meta-Analysis in R](#) using the meta package

Correlation and Regression

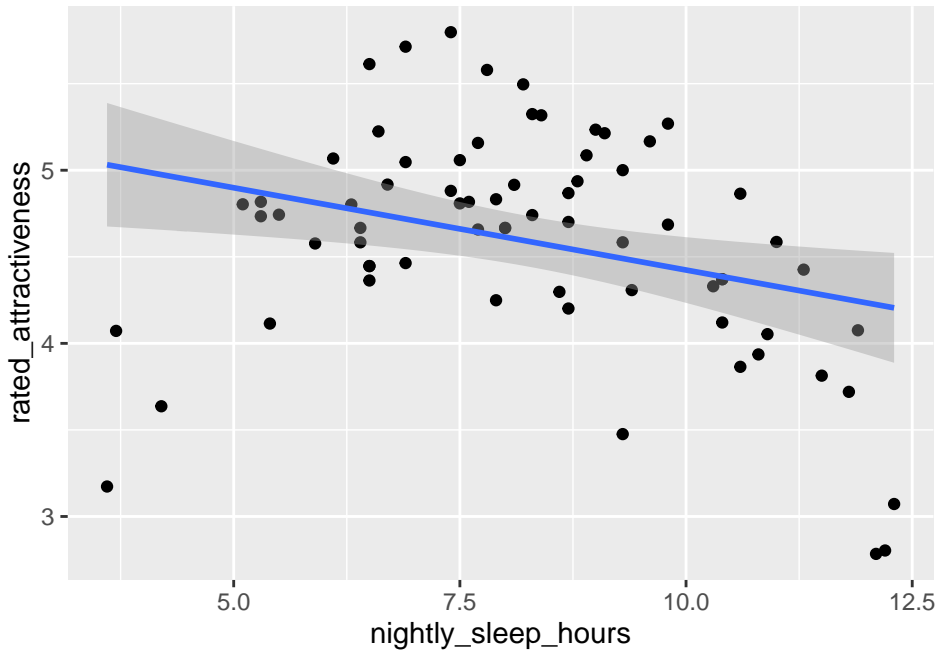
Scatterplots

The [Cookbook for R website](#) illustrates how to create scatterplots using the ggplot2 package.

```

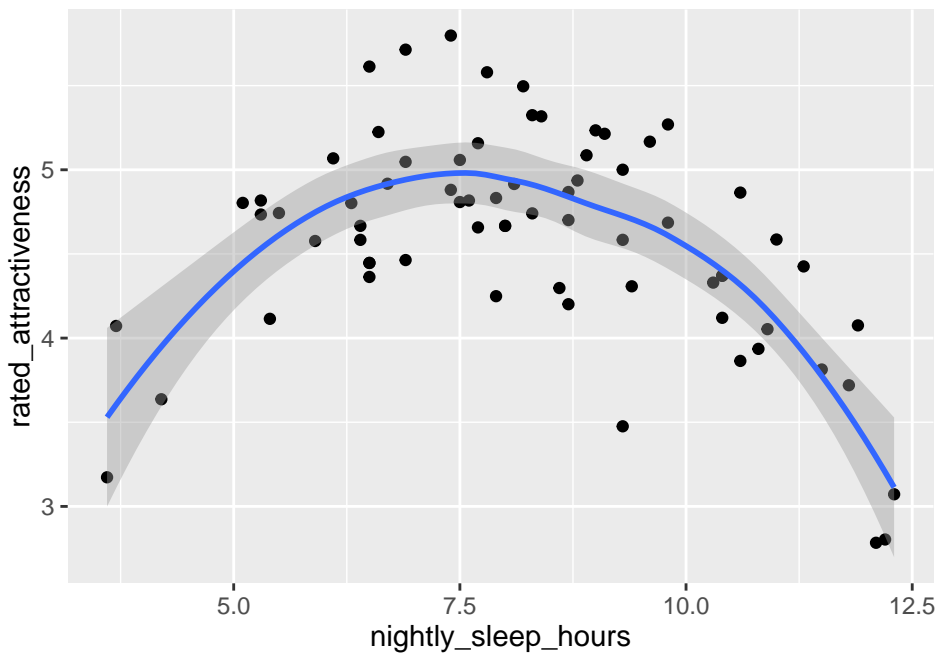
library(ggplot2)
ggplot(sleep_beauty, aes(x = nightly_sleep_hours, y = rated_attractiveness)) +
  geom_point() +
  geom_smooth(method = lm)    # Add linear regression line

```



Instead of a linear regression line, it is possible to add a non-linear regression (also known as a 'smoother') line to the plot. For the sleep_beauty data set, the non-linear regression line appears to fit the data better than the linear regression line.

```
ggplot(sleep_beauty, aes(x = nightly_sleep_hours, y = rated_attractiveness)) +  
  geom_point() +           # Use hollow circles  
  geom_smooth()            # Add nonlinear regression line
```



Correlation

The inbuilt R function `cor.test()` computes correlation coefficients and confidence intervals.

```
cor.test(sleep_beauty$nightly_sleep_hours, sleep_beauty$rated_attractiveness)
```

```
##
## Pearson's product-moment correlation
##
## data: sleep_beauty$nightly_sleep_hours and sleep_beauty$rated_attractiveness
## t = -2.7175, df = 68, p-value = 0.008337
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
## -0.51041935 -0.08420143
## sample estimates:
## cor
## -0.312983
```

Regression

The inbuilt R function `lm()` fits ordinary least-squares regression models. `confint()` returns confidence intervals for the regression coefficients, and `anova()` the ANOVA F-test.

```
fit <- lm(rated_attractiveness ~ nightly_sleep_hours, data = sleep_beauty)
summary(fit)
```

```
##
## Call:
## lm(formula = rated_attractiveness ~ nightly_sleep_hours, data = sleep_beauty)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.85879 -0.27999  0.03998  0.38011  1.12675
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)      5.37388    0.29805   18.030 < 2e-16 ***
## nightly_sleep_hours -0.09502    0.03497   -2.717  0.00834 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6121 on 68 degrees of freedom
## Multiple R-squared:  0.09796,    Adjusted R-squared:  0.08469
## F-statistic: 7.385 on 1 and 68 DF,  p-value: 0.008337
```

```
anova(fit)
```

```
## Analysis of Variance Table
##
## Response: rated_attractiveness
##              Df Sum Sq Mean Sq F value    Pr(>F)
## nightly_sleep_hours  1  2.7666  2.76661    7.3845 0.008337 **
## Residuals          68 25.4761  0.37465
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
confint(fit)
```

```
##                2.5 %      97.5 %  
## (Intercept)      4.7791288  5.96862811  
## nightly_sleep_hours -0.1647994 -0.02524601
```

The [Learn by Marketing](#) and [Harvard Regression Models in R](#) websites contain further information about how to conduct basic regression analyses in R. DataCamp also have [paid online courses](#) about correlation and regression analyses in R.

In addition to the `lm()` function built into R, there are numerous other functions and packages dedicated to fitting regression models. Probably the most famous is the `car` (Companion to Applied Regression) package, which is described in John Fox and Sanford Weisberg's book [An R Companion to Applied Regression](#).

Categorical Data - Frequencies, Proportions, Risk Ratios and Risk Differences

The `PropCIs` and `pairwiseCI` packages contains numerous functions for computing confidence intervals for single, paired and independent proportions. See also the `BinomCI()`, `BinomDiffCI()` and `BinomRatioCI()` functions in the [DescTools](#) package and the [R manual to accompany Agresti's Categorical Data Analysis](#) by Laura Thompson.

There are many R packages that include a function for computing the Chi-Square test - such as the `chisq_test()` function in the `coin` package.

There is also a package called `vcd` for visualising categorical data.

Extended Designs - One-Way and Factorial Designs

See the [Quick R website](#) for some basic examples of how to fit one-way and factorial models in R using the inbuilt `aov()` function. There is a more detailed discussion with examples [here](#).

If you are analysing data from these designs you may also find the `ez` package useful, as it is designed to provide a simplified interface for analysis of variance models.

There are several R packages that can be used for comparisons and contrasts, such as `contrast`, `multicon` and `lsmeans`.

Robust Methods

The `WRS2` package contains a collection of robust methods, including methods for computing effect sizes and confidence intervals for independent groups and repeated-measures designs. For example, the `yuen()` function can be used to compare two independent groups using trimmed means. The function returns the difference in trimmed means, a confidence interval, and p-value. By default, 20% trimming is used.

```
# Install package if not yet installed  
# install.packages("WRS2")  
library(WRS2)  
yuen(transcription ~ group, data = pen_laptop1)
```

```
## Call:  
## yuen(formula = transcription ~ group, data = pen_laptop1)  
##  
## Test statistic: 3.5543 (df = 32.41), p-value = 0.00119  
##  
## Trimmed mean difference: 5.07632  
## 95 percent confidence interval:  
## 2.1686      7.9841
```


The [WRS2 vignette](#) describes how to use the package.

Many additional robust statistics functions can be downloaded from [Rand Wilcox's website](#). The functions are described in [Professor Wilcox's books](#).

Summary

In this guide we have provided a brief overview of how to install R, the `itns` data package, and a variety of R packages and functions that will help get you started using the 'new statistics' in R. In addition to the packages covered in this guide, there are many others that are useful for data analysis - at the time of writing there were nearly [9000 available on CRAN](#). A good way to keep up to date with new packages and developments in the R community is to subscribe to [R bloggers](#). We wish you luck in your adventures using R - may your confidence intervals be short!

Reference

Cumming, G., & Calin-Jageman, R. (2017). *Introduction to the New Statistics*. New York; Routledge.