Methods in Ecology and Evolution

British Ecological Society

Methods in Ecology and Evolution 2016, **7**, 323–330 doi: 10.1111/2041-210X.12472

APPLICATION

Facilitating systematic reviews, data extraction and meta-analysis with the METAGEAR package for R

Marc J. Lajeunesse*

Department of Integrative Biology, University of South Florida, 4202 East Fowler Avenue, Tampa, FL 33620, USA

Summary

1. The R package ecosystem is rich in tools for the statistics of meta-analysis. However, there are few resources available to facilitate research synthesis as a whole.

2. Here, I present the METAGEAR package for R. It is a comprehensive, multifunctional toolbox with capabilities aimed to cover much of the research synthesis taxonomy: from applying a systematic review approach to objectively assemble and screen the literature, to extracting data from studies, and to finally summarize and analyse these data with the statistics of meta-analysis.

3. Current functionalities of METAGEAR include the following: an abstract screener GUI to efficiently sieve bibliographic information from large numbers of candidate studies; tools to assign screening effort across multiple collaborators/reviewers and to assess inter-reviewer reliability using kappa statistics; PDF downloader to automate the retrieval of journal articles from online data bases; automated data extractions from scatter-plots, box-plots and bar-plots; PRISMA flow diagrams; simple imputation tools to fill gaps in incomplete or missing study parameters; generation of random-effects sizes for Hedges' *d*, log response ratio, odds ratio and correlation coefficients for Monte Carlo experiments; covariance equations for modelling dependencies among multiple effect sizes (e.g. with a common control, phylogenetic correlations); and finally, summaries that replicate analyses and outputs from widely used but no longer updated meta-analysis software.

4. Research synthesis practices are vital to many disciplines in the sciences, including ecology and evolutionary biology, and METAGEAR aims to enrich the scope, quality and reproducibility of what can be achieved with the systematic review and meta-analysis of research outcomes.

Key-words: data extraction and retrieval, effect sizes, meta-analysis, MetaWin, quantitative reviews, research synthesis, systematic review

Introduction

Critical goals to research synthesis practices are to generate robust, aggregate views of accumulated research findings (Hunter & Schmidt 2004), and to explore sources of variability in that research (Lajeunesse 2010; Jennions, Lortie & Koricheva 2013a). Ideally, these goals are achieved by pairing systematic review methods with the statistics of meta-analysis to yield high-quality syntheses. For example, a systematic review is first used to carefully plan and implement a strategy to locate studies, assess their eligibility and extract relevant research findings in a repeatable and transparent way (Khan *et al.* 2003; Pullin & Stewart 2006). A meta-analysis is then used to statistically model sources of variability within and between studies with the aim to quantitatively weight and aggregate their findings (Koricheva, Gurevitch & Mengersen 2013). Although this taxonomy of practices is meant to produce scientifically explicit and defensible overviews of research, it is also accompanied by a variety of implementation challenges. For example, the number and complexity of tasks involved in generating a high-quality synthesis require a diverse combination of software tools – which are numerous and growing due to broad interdisciplinary interests in research synthesis (reviewed by Schmid *et al.* 2013). This diversity of tools can reduce efforts to make synthesis more repeatable and transparent, given that it can yield mixed standards for solving common problems with the screening, extraction and analysis of study outcomes (Bayliss & Beyer 2015).

Here, I introduce METAGEAR, an R package aimed at improving the reproducibility of systematic reviews and meta-analysis. It is a comprehensive toolbox that spans the entire research synthesis taxonomy, and supports a large diversity of functionalities to help screen abstracts, download articles, extract data from figures and model dependencies within and among effect sizes, and various other tools to improve the quality of statistics used in meta-analysis (Table 1). It is also meant to assist with the management of large collaborative projects by offering tools to help distribute work effort and to assess the reliability of that effort (Cohen 1960). Much of METAGEAR's toolbox arose from teaching research synthesis and participating in large projects where tasks are often distributed among multiple individuals with various skill sets, experience, access to

*Correspondence author. E-mail: lajeunesse@usf.edu

© 2015 The Author. Methods in Ecology and Evolution © 2015 British Ecological Society

Table 1. A sample of the R functions available in METAGEAR grouped by research synthesis taxonomy

Function	Description
Systematic review	
`abstract_screener()`	A GUI for vetting the titles and abstracts of study references
`effort_distribute()`	Distributes screening and extraction effort randomly and evenly among the reviewing team
`effort_merge()`	Combines effort from collaborative screening jobs and calculation of inter-reviewer agreement based on Cohen's kappa
`plot_PRISMA()`	Generates a flow diagram of the phases of a systematic review
Data retrieval and extraction	
`PDFs_collect()`	Automatically fetches PDFs using the DOI of multiple journal references
`figure_scatterPlot()`	Extracts data points from a scatter-plot figure image
`figure_boxPlot()`	Extracts means and error bars from a box-plot figure image
`figure_barPlot()`	Extracts means and error bars from a bar-plot figure image
`figure_add()`	Adds points manually to a figure image
Modelling effect sizes	
`impute_SD()`	Imputes missing standard deviations (SDs) using coefficient of variation of complete data or based on resampling techniques
`covariance_commonControl()`	Estimates the VCV matrix for multiple effect sizes with a common control
`covariance_multivariate()`	Estimates the VCV matrix based on a correlation matrix for multivariate effects
`random_d()`	Generates random-effect sizes based on Hedges' *d* or Cohen's *g*
`random_RR()`	Generates random log response ratios (RRs)
`random_r()`	Generates random Pearson product–moment correlation coefficients (*r*s)
`random_OR()`	Generates random log odd ratios (ORs)
Meta-analysis	
`replicate_MetaWin2.0()`	Duplicates the results and output of a meta-analysis performed by METAWIN 2.0 (Rosenberg, Adams & Gurevitch 2000)
`replicate_phyloMeta1.3()`	Replicates the results and output of traditional and phylogenetic meta-analyses performed by PHYLOMETA (Lajeunesse 2011b)
`MA_effectsTable()`	Generates an ANOVA-like-effects table first described by Hedges & Olkin (1985)
`effects_powerAnalysis()`	Computes the statistical power of a pooled effect size or Q-statistic

different computing platforms (e.g. Windows and Mac), and not having a single dedicated source that everyone could freely and remotely use to accomplish tasks. Given that clarity and repeatability are central themes to research synthesis, I hope that by packaging all these functionalities into a single source, I can set the groundwork for large and collaborative synthesis projects, as well as provide a standardized theatre for teaching systematic reviews and meta-analysis. Below, I outline stage-by-stage how METAGEAR facilitates research synthesis projects and promotes repeatability in their practices. As a complement to this outline, I also provide an R vignette with illustrative examples as Appendix S1.

SCREENING ABSTRACTS, DELEGATING TASKS AND RETRIEVING ARTICLES

Identifying candidate studies is one of the first key stages in a systematic review. However, literature searches with bibliographic data bases such as Web of Knowledge or Google Scholar will generate thousands of study references – this is unavoidable given that search terms should aim to yield the most inclusive results (see Curtis *et al.* 2013). The challenge here is that the title and abstract of all of these references need to be screened as a first attempt to sift those that are relevant for the synthesis project. There are already several options to help assist in reference screening and coding (i.e. tagging which to exclude/include). Reference-managing software such as Endnote or Mendeley can be shoehorned for this purpose (see King *et al.* 2011). There are also dedicated subscription-based screeners such as COVIDENCE (https://www.covidence.org) and DISTILLERSR (http://distillercer.com), and some even try to automate the screening process using machine learning (ABSTRACKR; Wallace *et al.* 2012). METAGEAR also supports an easy-to-use GUI screener to assist with the screening and coding of references (see Table 1 and Fig. 1). To help avoid potential selection bias based on the author, journal or year of the study (see Jennions *et al.* 2013b), the GUI offers only the title and abstract as the sole appraisal criteria for inclusion/exclusion (Fig. 1).

If screening tasks are to be delegated across a research team, METAGEAR supports collaborative tools (see Table 1) that can randomly divide study references among members evenly or unevenly (i.e. assigning more work to some and less to others) and generate dual-screening designs where two members independently code the same reference subset to evaluate the repeatability or consistency of screening tasks (also known as inter-reviewer agreement). Each member can then work remotely on their reference subset using the abstract screener. Screening tasks can also be redistributed should screening commitments change (or if team members fall behind, and it will happen!), and the teams' progress can finally be combined and summarized (see example of this workflow in the Appendix S1). For example, `effort_summary()` will calculate Cohen's (1960) kappa (K) to assess inter-reviewer agreement if a dual-reviewing design was implemented. Here is an example of this summary

based on the efforts from a screening exercise of 479 abstracts completed in my research synthesis course:

```
=== SCREENING EFFORT SUMMARY ===
149 candidate studies identified
141 studies excluded
189 studies with conflicting agreement needing
additional screening
—
479 TOTAL SCREENED
=== DUAL SCREENING DESIGN SUMMARY ===
```

Team_A	MAYBE	NO	YES	TOTAL	%	Team_B	MAYBE	NO	YES	TOTAL	%	Cohen's K
Bryan	27	44	27	98	20.46	Neal	11	44	43	98	20.46	0.2899 (fair)
Keith	15	28	56	99	20.67	Nick	10	52	37	99	20.67	0.3371 (fair)
Jake	20	28	51	99	20.67	Ashley	5	38	56	99	20.67	0.452 (moderate)
Neiman	1	57	41	99	20.67	Jeremy	11	53	35	99	20.67	0.5546 (moderate)
Marc	19	23	42	84	17.54	Jason	12	41	31	84	17.54	0.3585 (fair)
TOTAL	82	180	217	479	100.00	TOTAL	49	228	202	479	100.00	

The magnitude of Cohen's kappa is also scored following Landis & Koch's (1977) interpretation 'benchmarks' ranging from poor (<0·0), slight (0·0–0·2), fair (0·21–0·4), moderate (0·41–0·6), substantial (0·61–0·8), to almost perfect inclusion/exclusion agreement (0·81–1·0). References with disagreements for inclusion/exclusion should be discussed and rescreened by the team. In the example above, one of the reasons for the high disagreement was failure to establish clearly defined inclusion criteria prior to screening efforts. Curtis *et al.* (2013) provide helpful guidelines to avoid this issue by developing an explicit protocol for assessing the inclusion and exclusion of studies.

Once all study references have been screened, the next stage is to retrieve their full texts. This is tedious when there are hundreds or thousands of studies to collect. Here, METAGEAR can be used to automate the download of these studies (Table 1). Much like the reference-managing software ENDNOTE, METAGEAR's downloader uses the DOI (digital object identifier) of each bibliographic entry as a starting point to fetch PDFs online. Note that the DOIs of published articles are readily available and are often included when references are exported from bibliographic data bases (e.g. Web of Knowledge). Following the DOI link, the downloader then uses publisher-tailored HTML searches to identify PDFs to download – these *ad hoc* searches are necessary due to the lack of standardized access to e-journals. It is important to note that the success of these downloads will be conditional on the journal subscription coverage of the host institution running METAGEAR. However, this limitation can be harnessed to assess subscription or availability bias when collating studies from different types of institutions or countries. Another benefit of automating downloads is that it standardizes the file name of each downloaded PDF. The filenames of PDFs from online journals are often uninformative or cryptic, and having a standardized name (such as the unique study ID of each reference) will later help speed up the retrieval of documents for data extractions. Finally, it is also necessary to emphasize that the downloader is only meant to expedite the gathering of PDFs from published journal articles; the downloaded files will be incomplete, and additional (manual) effort will still be needed

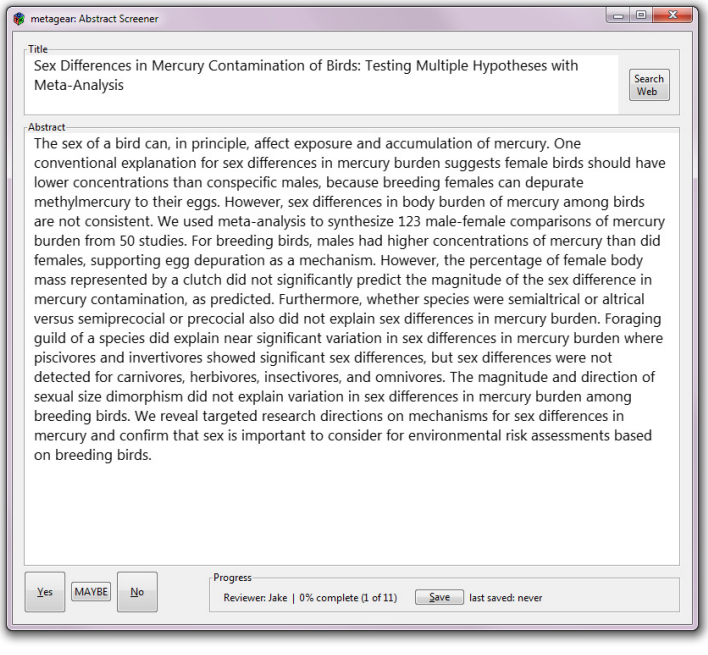


Fig. 1. The GUI from the `abstract_screener()` function to help sieve and code the references of candidate studies for systematic reviews and meta-analysis. Sometimes bibliographic data bases such as Web of Knowledge will not have complete bibliographic information for a journal article (e.g. DOI, abstract), and the title-to-browser button is meant to quickly search for this information on Google. The GUI also includes a progression tracker for the number of abstracts screened, as well as a button to save screening progress.

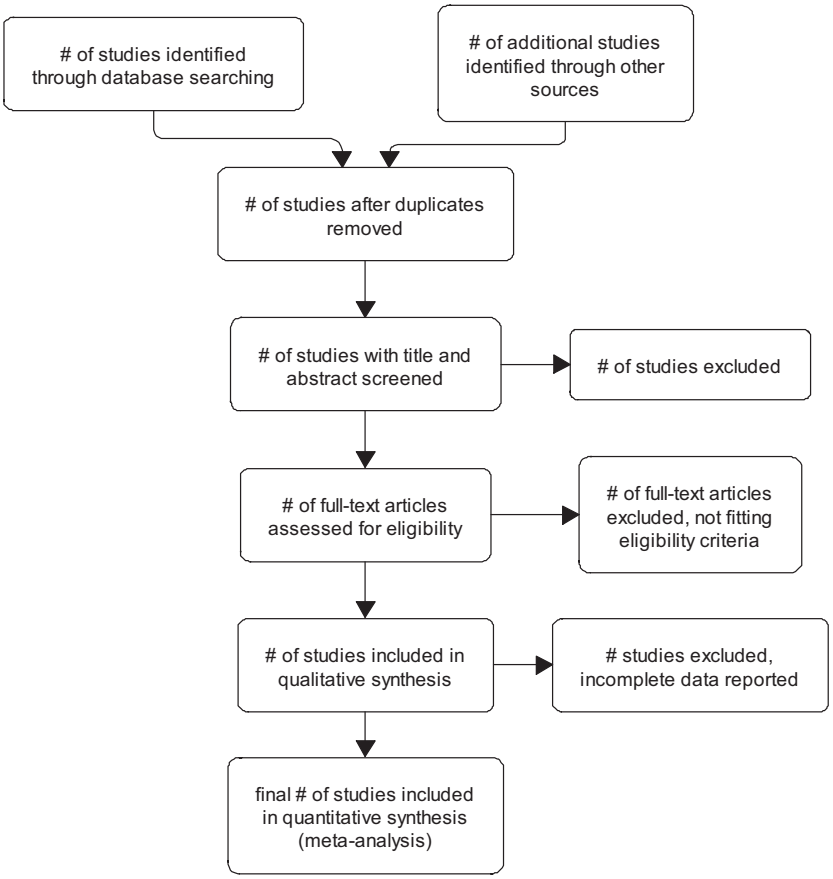


Fig. 2. An example of a PRISMA flow diagram generated by the `plot_PRISMA()` function.

to acquire all the full texts (e.g. articles that could not be downloaded, book chapters, dissertations). However, any download failure will get coded and summarized to further help locate these missing texts (see Appendix S1).

Finally, it is often useful to record and present each phase of the screening process and number of studies included and excluded in the final meta-analysis with a PRISMA diagram (Preferred Reporting Items for Systematic Reviews and Meta-Analyses; see Moher *et al.* 2009). The `plot_PRISMA()` function can be used to quickly generate and update flow diagrams as each phase or stage gets completed. For example, below is the R script to generate the plot presented in Fig. 2:

```
phases c("START_PHASE: # of studies identified
through database searching",
"START_PHASE: # of additional studies identified
through other sources",
"# of studies after duplicates removed",
"# of studies with title and abstract screened",
"EXCLUDE_PHASE: # of studies excluded",
"# of full-text articles assessed for eligibility",
"EXCLUDE_PHASE: # of full-text articles excluded,
not fitting eligibility criteria",
"# of studies included in qualitative synthesis",
"EXCLUDE_PHASE: # studies excluded, incomplete data
reported",
"final # of studies included in quantitative
synthesis (meta-analysis)")
plot_PRISMA(phases)
```

EXTRACTING AND IMPUTING DATA

Extracting data from studies to calculate effect sizes (a common currency that quantitatively summarizes the magnitude and sign of study outcomes) is by far the most difficult and time-consuming stage of the entire research synthesis process. It is a manual activity where researchers must read and interpret the text/tables/figures of each study with the goals of unearthing all the quantitative data needed to calculate effect sizes (such as sample sizes, means; Hedges 1981; Lajeunesse & Forbes 2003), and all the qualitative/quantitative information needed to establish moderator groups and annotations useful for hypothesis testing and quality control (see Koricheva, Gurevitch & Mengersen 2013). This activity is never straightforward, and there are many challenges that can make studies nearly impenetrable for data extractions.

A particularly challenging task is extracting or reverse engineering the plotted data from published figures. Although there are many options to help extract data available only in graphical form, such as general image manipulation software (i.e. MS POWERPOINT, MS PAINT), image software with specialized plug-ins (IMAGEJ; Abramoff, Magelhaes & Ram 2004), or dedicated data extraction software such as DATATHIEF (Tummers 2006) or GRAPHCLICK (In Neuchatel 2008), these tools are often not cross-platform and can be poorly documented. Further, none of these are tailored with research synthesis in mind. METAGEAR offers manual and automated tools to help extract numerical data

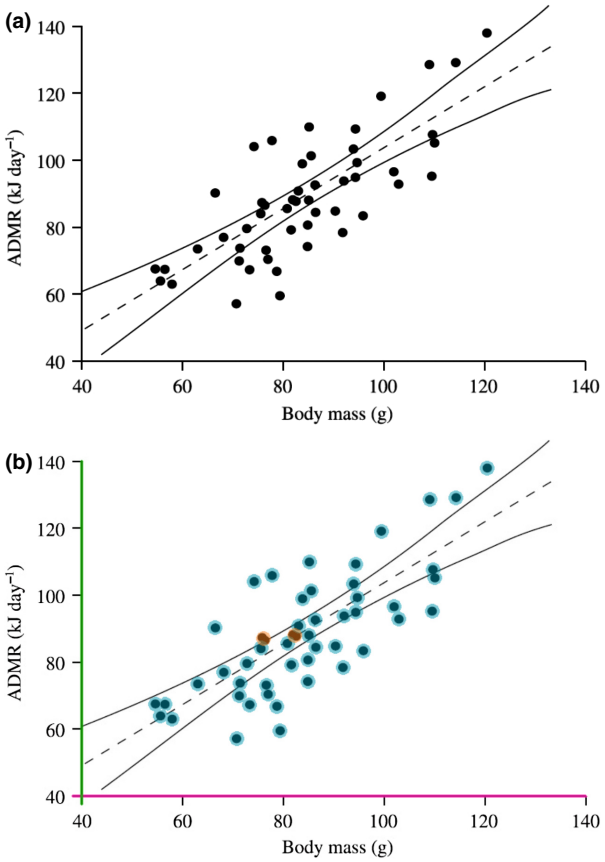


Fig. 3. An example of automated data extractions using METAGEAR from a figure image using `figure_scatterPlot()`. The top panel (a) is the original figure reported by Kam *et al.* (2003; Fig. 2). The bottom panel (b) contains the detected objects on that figure: blue circles are the detected points, orange circles are clusters of points that could not be separated, green vertical line is the detected *Y*-axis, and the pink horizontal line is the detected *X*-axis. These detected objects are used to estimate the regression line and correlation coefficient of the points depicted in the image.

from scatter-plot, box-plot and bar-plot figures. Below is a worked example where I use METAGEAR to extract the plotted points from Fig. 2 of Kam *et al.*'s (2003) study on predictors of metabolic rates in mice (this figure is reprinted here as Fig. 3a). Kam *et al.*'s (2003) study also reports the regression coefficients and R^2 of this plot. I will compare these reported values with those estimated by METAGEAR (the original published results were as follows: $Y = 12{\cdot}03 + 0{\cdot}907{*}X$ with an $R^2 = 0{\cdot}59$ and a sample size of $N = 51$).

I first obtained an image of the figure directly from the article's PDF with a screenshot, then cropped it to include only the figure itself and its axes and finally saved it as a JPG (file name: `"Kam_et_al_2003_Fig 2.jpg"`; see Fig. 3a). Obtaining an image directly from a PDF is often preferred over images provided by online HTML articles as they tend to be higher quality and can be resized without quality loss. High image quality will improve the success of automated data extractions as it increases the detection and separation of points and axes (Pau *et al.* 2010). The function `figure_scatterPlot()` can then be used to extract the data from the image as follows:

```
>figure_scatterPlot("Kam_et_al_2003_Fig2.jpg")
```

The default use of this function will plot a raster image of all the detected axes and scatter-plot points (see Fig. 3b), return an R data frame of the estimated *X* and *Y* data from this plot (which can be analysed, plotted, saved or rescaled) and print to console the following regression summary and estimated effect size and variance (here the effect size is Pearson's *r* correlation coefficient):

```
regression fit: Y = 11·92586 + 0·9077 * X, R-squared =
0·59498
Pearsons r = 0·7713478, var(r) = 0·0034903, N = 49
```

The extracted regression parameters are very similar to what was reported by Kam *et al.* (2003).

However, note that `figure_scatterPlot()` was unable to separate two clusters of points on the reported figure. These clusters are emphasized in orange (see Fig. 3b), and although not shown, they were also coded along with the *X* and *Y* data frame of the extracted data. This is also why the sample size of the extracted regression was $N = 49$ and not 51 as reported by Kam *et al.* (2003). I could not find any established guidelines on how to treat clusters of points that cannot be separated within a figure, but given that this issue will be common for scatter-plots with high *N* and data densities, METAGEAR supports two options. First, it will automatically treat a cluster as a single point based on the centroid of that cluster. This is perhaps analogous to imputing the missing points with a single estimate based on the average *X* and *Y* within that cluster. In the example above, this imputation of clusters yielded a very similar regression outcome as those reported in the original study. Secondly, METAGEAR also offers a way to manually interpolate individual points within a plot with the `figure_addPoints()` function. Here, the user essentially points and clicks on the figure to add points not detected by the extraction algorithm. As a final point, the defaults of `figure_scatterPlot()` are optimized for images of size ca. 650 by 650 pixels; tweaking the default parameters such as axis-bar thickness and point size will be necessary for successful extractions from images outside this size range (see examples in Appendix S1).

Another common challenge when extracting data from studies is when crucial information for computing effect sizes is neither found nor reported in the study – such as the variances, standard deviations (SDs) or standard errors (SEs). This is a widespread barrier for calculating effect sizes (Lajeunesse & Forbes 2003), but one common workaround is to use conversions or alternative formulations of effect sizes to compute effects (reviewed by Lipsey & Wilson 2001; Lajeunesse 2013a). A large diversity of conversions are available (i.e. converting *t*- or *F*-tests to Hedges' *d*), and many are offered by the COMPUTE.ES R package (Del Re 2012) and Lipsey & Wilson's (2001) companion website: http://www.campbellcollaboration.org/resources/effect_size_input.php. When conversions are not possible, METAGEAR supports a few simple imputation

approaches that can be used to fill gaps in missing SDs (see Table 1). These imputation tools were outlined by Lajeunesse (2013a) and either involve estimating the coefficient of variation from the (complete) observed data or rely on resampling approaches to fill gaps. Therefore, as a caveat to their use and success, these imputation tools should only be applied when data extractions from all studies have been completed. These SD imputation tools include Bracken's (1992) method for filling missing information using the coefficient of variation from all studies with complete information. METAGEAR also provides two variations on Rubin & Schenker's (1991) 'hot deck' imputation approach. The first is a strictly random hot deck imputation where SDs are imputed using all the observed SDs, while the other imputes only SDs that are nearest neighbours relative to their means (i.e. impute SDs from data with means of similar scale).

MODELLING EFFECT SIZES AND DEPENDENCIES

There is considerable interest in modelling dependencies within and among effect sizes for meta-analysis (Gleser & Olkin 1994; Curtis & Queenborough 2012). Dependent effect sizes violate some of the most basic assumptions of statistics, and the goal is to minimize inferential errors when hypothesis testing with meta-analysis (Lajeunesse 2009; Mengersen, Jennions & Schmid 2013). Dependencies can range from using shared information to calculate multiple effect sizes, to modelling effect sizes in a phylogenetic context. METAGEAR supports a broad diversity of tools to calculate covariances and variance–covariance matrices for dependent effect sizes (Table 1). One pervasive form is when multiple effect sizes share data from a common control group (see example in Lajeunesse 2011a). METAGEAR offers the covariance equations needed to model this common-control problem for all the major effect size metrics (e.g. Hedges' *d*, log response ratio, odds ratio, correlation coefficient) and will also generate the variance–covariance matrices needed to integrate these covariances into meta-analysis using the METAFOR R package (Viechtbauer 2010). An example of this application is found in the Appendix S1. Similarly, phylogenetic dependencies assuming simple Brownian motion models of phenotypic evolution can be modelled in METAGEAR similar to analyses produced by console program PHYLOMETA (Lajeunesse 2011b; Chamberlain *et al.* 2012; Lajeunesse, Rosenberg & Jennions 2013). Finally for Monte Carlo modelling, METAGEAR has random number generators for several of the major effect size metrics used in meta-analysis (Table 1). These generators create random-effect sizes by sampling their known probability distribution when possible (e.g. non-central *t*-distribution for Hedges' *d*) or generate random study parameters (i.e. means, SD and *N*) that can be used to compute random-effect sizes.

META-ANALYSIS AND THE REPRODUCIBILITY OF SYNTHESES

The most basic goals of meta-analysis software are to provide weighted regression tools for combining and comparing effect sizes (i.e. with the inverse variance of each effect size as weights; Hedges & Olkin 1985), and to model sources of statistical heterogeneity among effect sizes that exist beyond sampling error, such as the between-study variance (τ^2) component of random-effects models (Higgins & Thompson 2002). Ecologists and evolutionary biologists have historically relied heavily on METAWIN (Rosenberg, Adams & Gurevitch 2000) to meet these statistical goals. With more than 950 published applications to date (Google Scholar: 7/08/15), and with nearly 25% of these from the last 2 years alone (2013–2015), there is also little evidence that this software is losing traction among users. This popularity is also remarkable considering the availability of more sophisticated meta-analysis software, such as COMPREHENSIVE META-ANALYSIS (reviewed by Schmid *et al.* 2013) and the free OPENMEE (Wallace *et al.* 2015), as well as the METAFOR R package which now dominates the statistical analyses of many disciplines (Viechtbauer 2010). There is a lot that could be said about creating easy-to-use, GUI-driven, discipline-tailored software such as METAWIN.

Unfortunately, METAWIN is no longer maintained (last updated in 2007), and it is important that analyses from this software can be reproduced for historical compatibilities as new statistical developments and practices emerge. Further, duplicating the analyses of METAWIN may not be straightforward in R given that some of its statistical approaches were

```
theData <- read.csv(file = "effect.dta", header = TRUE)
replicate_MetaWin2.0(Effect ~ 1, weights = Var, data = theData)

=== START of Rosenberg et al. (2002) metaWin 2.0 output ===

Estimate of pooled variance: 0.243866

SUMMARY RESULTS

Heterogeneity        df    Prob(Chi-Square)
-------------------------------------------------
Qtotal  28.729215    29    0.47923

Mean Effect Size        95% CI                  Bootstrap CI            Bias CI
-------------------------------------------------------------------------------------------
E++  0.916966           0.621266 to 1.212667    0.622620 to 1.196920   0.606058 to 1.187931

Sqrt Pooled Variance = 0.493828
 Mean Study Variance = 0.5015  Ratio = 0.984702

-------------------------------------------------

=== END of Rosenberg et al. (2002) MetaWin 2.0 output ===
```

never fully documented. For example, it uses a *t*-distribution to estimate 95% confidence intervals and sums of squares of regression models as Q-tests (following Hedges & Olkin 1985) applies a DerSimonian & Laird (1986) approach to estimating τ^2, and when a moderator variable is included in a random-effects analysis, it recycles the τ^2 from that model to estimate an overall (grand mean) pooled effect size. The function `replicate_MetaWin2.0()` aims to duplicate these statistical approaches and the way they were reported. Below, using one of the original data sets '`effect.dta`' packaged with METAWIN 2.0, is the output from a random-effects meta-analysis based on this R function:

Conclusions and prospectus

Looking to the future, there is still room to grow in terms of developing tools for research synthesis – such as automating PDF annotations, text-mining tools to enhance data retrieval and extraction, methods to estimate statistical power (Lajeunesse 2013b), multiple-imputation tools (Lajeunesse 2013a; Ellington *et al*. 2015) and better approaches to assess publication bias. There is also a continuing need to improve how synthesis results and meta-analytic outcomes are presented and reported in research articles (Lortie *et al*. 2015). Transparency and clarity are also crucial to strengthen the repeatability of synthesis findings, but are also necessary to improve the second-order analysis of these synthesis-level outcomes (Hunter & Schmidt 2004). Finally, given the often overwhelming interdisciplinary nature of research synthesis methods, I hope that by offering a single dedicated source for all these tools, I can facilitate the training, education and adoption of these diverse methodologies and elevate the standards and practices of what can be achieved by ecologists and evolution biologists with these tools.

Acknowledgements

I thank J. Richardson, J. Zydek, N. Ogburn, B. MacNeill, J. Zloty and three anonymous reviewers for comments and suggestions. I also thank my colleagues in the OPENMEE software team, J. Gurevitch and B. Wallace, for persuading me to develop tools in R. Funding was supported by NSF grants DBI-1262545 and DEB-1451031.

Documentation and availability

The website and updated vignettes (Appendix S1) for METAGEAR can be found at https://lajeunesse.myweb.usf.edu. Documentation and source code are freely available on CRAN (http://cran.r-project.org/web/packages/metagear) and GITHUB (https://github.com/cran/metagear).

References

Abramoff, M.D., Magelhaes, P.J. & Ram, S.J. (2004) Image processing with ImageJ. *Biophotonics International*, **11**, 36–42.

Bayliss, H.R. & Beyer, F.R. (2015) Information retrieval for ecological syntheses. *Research Synthesis Methods*, **6**, 136–148.

Bracken, M.B. (1992) Statistical methods for analysis of effects of treatment in overviews of randomized trials. *Effective Care of the Newborn Infant* (eds J.C. Sinclair & M.B. Bracken), pp. 13–20. Oxford University Press, Oxford.

Chamberlain, S.A., Hovick, S.M., Dibble, C.J., Rasmussen, N.L., Van Allen, B.G., Maitner, B.S. *et al*. (2012) Does phylogeny matter? Assessing the impact of phylogenetic information in ecological meta-analysis. *Ecology Letters*, **15**, 627–636.

Cohen, J. (1960) A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, **20**, 37–46.

Curtis, P.S. & Queenborough, S.A. (2012) Raising the standards for ecological meta-analysis. *New Phytologist*, **195**, 279–281.

Curtis, P.S., Mengersen, K., Lajeunesse, M.J., Rothstein, H.R. & Stewart, G.B. (2013) Extraction and critical appraisal of data. *Handbook of Meta-Analysis in Ecology and Evolution* (eds J. Koricheva, J. Gurevitch & K. Mengersen), pp. 52–60. Princeton University Press, Princeton.

Del Re, A.C. (2012) R package "compute.es". Available at: http://cran.rproject.org/package = compute.es [accessed 30 December 2014].

DerSimonian, R. & Laird, N. (1986) Meta-analysis in clinical trials. *Controlled Clinical Trials*, **7**, 177–188.

Ellington, E.H., Bastille-Rousseau, G., Austin, C., Landolt, K.N., Pond, B.A., Rees, E.E., Robar, N. & Murray, D.L. (2015) Using multiple imputation to estimate missing data in meta-regression. *Methods in Ecology and Evolution*, **6**, 153–163.

Gleser, L.J. & Olkin, I. (1994) Stochastically dependent effect sizes. *The Handbook of Research Synthesis* (eds H.M. Cooper & L.V. Hedges), pp. 339–355. Russell Sage Foundation, New York.

Hedges, L.V. (1981) Distribution theory for Glass's estimator of effect size and related estimators. *Journal of Educational Statistics*, **6**, 107–128.

Hedges, L.V. & Olkin, I. (1985) *Statistical Methods for Meta-Analysis*. Academic Press, New York, NY, USA.

Higgins, J.P.T. & Thompson, S.G. (2002) Quantifying heterogeneity in a meta-analysis. *Statistics in Medicine*, **21**, 1539–1558.

Hunter, J.E. & Schmidt, F.L. (2004) *Methods of Meta-Analysis: Correcting for Error and Bias in Research Findings*. Sage, Newbury Park.

In Neuchatel, C.H. (2008). *GraphClick* (Version 3.0). Arizona-Software. Available at: http://www.arizona-software.ch [accessed 30 December 2014].

Jennions, M.D., Lortie, C.J. & Koricheva, J. (2013a) Using meta-analysis to test ecological and evolutionary theory. *Handbook of Meta-Analysis in Ecology and Evolution* (eds J. Koricheva, J. Gurevitch & K. Mengersen), pp. 381–403. Princeton University Press, Princeton.

Jennions, M.D., Lortie, C.J., Rosenberg, M.S. & Rothstein, H.R. (2013b) Publication and related biases. *Handbook of Meta-Analysis in Ecology and Evolution* (eds J. Koricheva, J. Gurevitch & K. Mengersen), pp. 207–236. Princeton University Press, Princeton.

Kam, M., Cohen-Gross, S., Khokhlova, I.S., Degen, A.A. & Geffen, E. (2003) Average daily metabolic rate, reproduction and energy allocation during lactation in the Sundevall jird *Meriones crassus*. *Functional Ecology*, **17**, 496–503.

Khan, K.S., Kunz, R., Kleijnen, J. & Antes, G. (2003) Five steps to conducting a systematic review. *Journal of the Royal Society of Medicine*, **96**, 118–121.

King, R., Hooper, B. & Wood, W. (2011) Using bibliographic software to appraise and code data in educational systematic review research. *Medical Teacher*, **33**, 719–723.

Koricheva, J., Gurevitch, J. & Mengersen, K. (2013) *Handbook of Meta-Analysis in Ecology and Evolution*. Princeton University Press, Princeton.

Lajeunesse, M.J. (2009) Meta-analysis and the comparative phylogenetic method. *American Naturalist*, **174**, 369–381.

Lajeunesse, M.J. (2010) Achieving synthesis with meta-analysis by combining and comparing all available studies. *Ecology*, **91**, 2561–2564.

Lajeunesse, M.J. (2011a) On the meta-analysis of response ratios for studies with correlated and multi-group designs. *Ecology*, **92**, 2049–2055.

Lajeunesse, M.J. (2011b) phyloMeta: a program for phylogenetic comparative analyses with meta-analysis. *Bioinformatics*, **27**, 2603–2604.

Lajeunesse, M.J. (2013a) Recovering missing or partial data from studies: a survey of conversions and imputations for meta-analysis. *Handbook of Meta-Analysis in Ecology and Evolution* (eds J. Koricheva, J. Gurevitch & K. Mengersen), pp. 195–206. Princeton University Press, Princeton.

Lajeunesse, M.J. (2013b) Power statistics for meta-analysis: test for mean effects and homogeneity. *Handbook of Meta-Analysis in Ecology and Evolution* (eds J. Koricheva, J. Gurevitch & K. Mengersen), pp. 348–363. Princeton University Press, Princeton.

Lajeunesse, M.J. & Forbes, M.R. (2003) Variable reporting and quantitative reviews: a comparison of three meta-analytical techniques. *Ecology Letters*, **6**, 448–454.

Lajeunesse, M.J., Rosenberg, M.R. & Jennions, M.D. (2013) Phylogenetic nonindependence and meta-analysis. *Handbook of Meta-Analysis in Ecology and Evolution* (eds J. Koricheva, J. Gurevitch & K. Mengersen), pp. 284–299. Princeton University Press, Princeton.

Landis, J.R. & Koch, G.G. (1977) The measurement of observer agreement for categorical data. *Biometrics*, **33**, 159–174.

Lipsey, M.W. & Wilson, D.B. (2001) *Practical Meta-Analysis*. Sage Publications, Thousand Oaks.

Lortie, C.J., Stewart, G., Rothstein, H. & Lau, J. (2015) How to critically read ecological meta-analyses. *Research Synthesis Methods*, **6**, 124–133.

Mengersen, K., Jennions, M.D. & Schmid, M.D. (2013) Statistical models for the meta-analysis of nonindependent data. *Handbook of Meta-Analysis in Ecology and Evolution* (eds J. Koricheva, J. Gurevitch & K. Mengersen), pp. 255–283. Princeton University Press, Princeton.

Moher, D., Liberati, A., Tetzlaff, J., Altman, D.G. & PRISMA Group. (2009) Preferred reporting items for systematic reviews and meta-analyses: the PRISMA statement. *BMJ*, **339**, b2535.

Pau, G., Fuchs, F., Sklyar, O., Boutros, M. & Huber, W. (2010) EBImage–an R package for image processing with applications to cellular phenotypes. *Bioinformatics*, **26**, 979–981.

Pullin, A.S. & Stewart, G.S. (2006) Guidelines for systematic review in conservation and environmental management. *Conservation Biology*, **20**, 1647–1656.

Rosenberg, M.S., Adams, D.C. & Gurevitch, J. (2000) *MetaWin: Statistical Software for Meta-Analysis*. Sinauer Associates, Sunderland.

Rubin, D.B. & Schenker, N. (1991) Multiple imputation in health-care databases: an overview and some applications. *Statistics in Medicine*, **10**, 585–598.

Schmid, C.H., Stewart, G.B., Rothstein, H.R., Lajeunesse, M.J. & Gurevitch, J. (2013) Software for statistical meta-analysis. *Handbook of Meta-Analysis in Ecology and Evolution* (eds J. Koricheva, J. Gurevitch & K. Mengersen), pp. 174–194. Princeton University Press, Princeton.

Tummers, B. (2006) *DataThief III*. Available at: http://datathief.org [accessed 30 December 2014].

Viechtbauer, W. (2010) Conducting meta-analyses in R with the metafor package. *Journal of Statistical Software*, **36**, 1–48.

Wallace, B.C., Small, K., Brodley, C.E., Lau, J. & Trikalinos, T.A. (2012) Deploying an interactive machine learning system in an evidence-based practice center: abstrackr. *Proceedings of the 2nd ACM SIGHIT International Health Informatics Symposium, Miami, Florida, USA*. New York: ACM, 819–824.

Wallace, B.C., Dietz, G., Lajeunesse, M.J., Dahabreh, I.J., Trikalinos, T.A., Schmid, C.H. & Gurevitch, J. (2015) OpenMEE: Intuitive, open-source software for meta-analysis in ecology and evolutionary biology. Available at: http://www.cebm.brown.edu/openmee [accessed 15 June 2015].

Received 22 May 2015; accepted 27 August 2015
Handling Editor: Richard Fitzjohn

Supporting Information

Additional Supporting Information may be found in the online version of this article.

Appendix S1. Basic examples of screening studies, extracting data, and meta-analysis with the metagear package for R.

Basic examples of screening studies, extracting data, and meta-analysis with the *metagear* package for R

Marc J. Lajeunesse

University of South Florida, August 5th 2015 (for metagear v. 0.2)

Introduction

The **metagear** package for R contains tools for facilitating systematic reviews, data extraction, and meta-analyses. It aims to facilitate research synthesis as a whole, by providing a single source for several of the common tasks involved in screening studies, extracting outcomes from studies, and performing statistical analyses on these outcomes using meta-analysis. Below are a few illustrative examples of applications of these functionalities.

Updates to these examples will be posted on our research webpage at USF.

For the source code of **metagear** see: http://cran.r-project.org/web/packages/metagear/index.html.

Delegating reference screening effort to a team

One of the first tasks of a systematic review is to screen the titles and abstracts of study references to assess their relevance for the synthesis project. For example, after a bibliographic search using Web of Science, there may be thousands of references generated; references from experimental studies, modeling studies, review papers, commentaries, etc. These need to be reviewed individually as a first pass to exclude those that do not fit the synthesis project; such as excluding simulation studies that do not report experimental outcomes useful for estimating an effect size.

However, individually screening thousands of references is time consuming, and large synthesis projects may benefit from delegating this screening effort to a research team. Having multiple people screen references also provides an opportunity to assess the repeatability of these screening decisions.

In this example, we have the following goals:

1. Initialize a dataframe containing bibliographic data (tile, abstract, journal) from multiple study references.
2. Distribute these references randomly to two team members.

3. Merge and summarize the screening efforts of this team.

First, let's start by loading and exploring the contents of a pre-packaged dataset from **metagear** that contains the bibliographic information of 11 journal articles (`example_references_metagear`). These data are a subset of references generated from a search in Web of Science for "Genome size", and contain the abstracts, titles, volume, page numbers, and authors of these references.

```
# load package
library(metagear)
# load a bibliographic dataset with the authors, titles, and abstracts of multiple study references
data(example_references_metagear)
# display the bibliographic variables in this dataset
names(example_references_metagear)

## [1] "AUTHORS"  "YEAR"     "TITLE"    "JOURNAL"  "VOLUME"   "LPAGES"   "UPAGES"   "DOI"      "ABSTRACT"

# display the various Journals that these references were published in
example_references_metagear["JOURNAL"]

##                                              JOURNAL
## 1  BIOCHEMICAL AND BIOPHYSICAL RESEARCH COMMUNICATIONS
## 2                          EVOLUTIONARY ECOLOGY RESEARCH
## 3                                    AMERICAN NATURALIST
## 4                                                   GENE
## 5                                           VIRUS GENES
## 6                          JOURNAL OF SHELLFISH RESEARCH
## 7                        JOURNAL OF GENERAL MICROBIOLOGY
## 8                                    APPLIED GEOCHEMISTRY
## 9      JOURNAL OF DRUG DELIVERY SCIENCE AND TECHNOLOGY
## 10                                    BIOLOGIA PLANTARUM
## 11                                           GENOMICS
```

Our next step is to initialize/prime this dataset for screening tasks. Our goal is to distribute screening efforts to two screeners/reviewers: "Christina" and "Luc". Here each reviewer will screen a separate subset of these references (a forthcoming example will review how to set up a dual screening design where each member screens the same references). The dataset first needs to be initialized as follows:

```
# prime the study-reference dataset
theRefs <- effort_initialize(example_references_metagear)
# display the new columns added by effort_initialize
names(theRefs)

##  [1] "STUDY_ID"  "REVIEWERS" "INCLUDE"   "AUTHORS"   "YEAR"      "TITLE"     "JOURNAL"   "VOLUME"
"LPAGES"    "UPAGES"    "DOI"       "ABSTRACT"
```

Note that the `effort_initialize()` function added three new columns: "STUDY_ID" which is a unique number for each reference (e.g., from 1 to 11), "REVIEWERS" an empty column with NAs that will be later populated with our reviewers (e.g., Christina and Luc), and finally the "INCLUDE" column, which will later contain the screening efforts by the two reviewers.

Screening efforts are essentially how individual study references get coded for inclusion in the synthesis project; currently the "INCLUDE" column has each reference coded as "not vetted", indicating that each reference has yet to be screened.

Our next task is to delegate screening efforts to our two reviewers Christina and Luc. Our goal is to randomly distribute these references to each reviewer.

```r
# randomly distribute screening effort to a team
theTeam <- c("Christina", "Luc")
theRefs_unscreened <- effort_distribute(theRefs, reviewers = theTeam)
# display screening tasks
theRefs_unscreened[c("STUDY_ID", "REVIEWERS")]
```

```
##    STUDY_ID REVIEWERS
## 1         1       Luc
## 2         2 Christina
## 3         3 Christina
## 4         4 Christina
## 5         5       Luc
## 6         6 Christina
## 7         7       Luc
## 8         8       Luc
## 9         9 Christina
## 10       10 Christina
## 11       11       Luc
```

The screening efforts can also be delegated unevenly, such as below where Luc will take on 80% of the screening effort:

```r
# randomly distribute screening effort to a team, but with Luc handeling 80% of the work
theRefs_unscreened <- effort_distribute(theRefs, reviewers = theTeam, effort = c(20, 80))
theRefs_unscreened[c("STUDY_ID", "REVIEWERS")]
```

```
##    STUDY_ID REVIEWERS
## 1         1       Luc
## 2         2       Luc
## 3         3       Luc
## 4         4       Luc
## 5         5 Christina
## 6         6       Luc
## 7         7       Luc
## 8         8       Luc
## 9         9       Luc
## 10       10 Christina
## 11       11       Luc
```

The effort can also be redistributed with the `effort_redistribute()` function. In the above example we assigned Luc 80% of the work. Now let's redistribute half of Luc's work to a new team member "Patsy".

```r
theRefs_Patsy <- effort_redistribute(theRefs_unscreened,
                                     reviewer = "Luc",
                                     remove_effort = "50", # move 50% of Luc's work to Patsy
                                     reviewers = c("Luc", "Patsy")) # team members loosing and picking up
work
theRefs_Patsy[c("STUDY_ID", "REVIEWERS")]
```

```
##    STUDY_ID REVIEWERS
## 5         5 Christina
## 10       10 Christina
## 1         1     Patsy
## 2         2     Patsy
## 3         3       Luc
## 4         4     Patsy
## 6         6       Luc
## 7         7       Luc
## 8         8       Luc
## 9         9       Luc
## 11       11     Patsy
```

The references have now been randomly assigned to either Christina or Luc. The whole initialization of the reference dataset with `effort_initialize()` can be abbreviated with

`effort_distribute(example_references_metagear, reviewers = c("Christina", "Luc"), initialize = TRUE)`.

Now that screening tasks have been distributed, the next stage is for reviewers to start the manual screening of each assigned reference. This is perhaps best done by providing a separate file of these references to Christina and Luc. They can then work on screening these references separately and remotely. Once the screening is complete, we can then merge these files into a complete dataset (we'll get to this later).

The `effort_distribute()` function can also save to file each reference subset; these can be given to Christina and Luc to start their work. This is done by setting the 'save_split' parameter to TRUE.

```
# randomly distribute screening effort to a team, but with Luc handling 80% of the work,
# but also saving these screening tasks to separate files for each team member
theRefs_unscreened <- effort_distribute(theRefs, reviewers = theTeam, effort = c(20, 80), save_split = TRUE)

## 2 files saved in: C:/Users/lajeunesse@usf.edu/Desktop

theRefs_unscreened[c("STUDY_ID", "REVIEWERS")]

##    STUDY_ID REVIEWERS
## 1         1       Luc
## 2         2 Christina
## 3         3       Luc
## 4         4       Luc
## 5         5 Christina
## 6         6       Luc
## 7         7       Luc
## 8         8       Luc
## 9         9       Luc
## 10       10       Luc
## 11       11       Luc

list.files(pattern = "effort")

## [1] "effort_Christina.csv" "effort_Luc.csv"
```

These two effort_*.csv files contain the assigned references for Christina and Luc. These can be passed on to each team member so that they can begin screening/coding each reference for inclusion in the synthesis project.

References should be coded as "YES" or "NO" for inclusion, but can also be coded as "MAYBE" if bibliographic information is missing or there is inadequate information to make a proper assessment of the study.

The `abstract_screener()` function can be used to facilitate this screening process (an example is forthcoming), but for the sake of introducing how screening efforts can be merged and summarized, I manually coded all the references in both of Christina's and Luc's effort_*.csv files. Essentially, I randomly coded each references as either "YES", "NO", or "MAYBE". These files now contain the completed screening efforts.

We can merge these two files with the completed screening efforts using the `effort_merge()` function, as well as summarize the outcome of screening tasks using the `effort_summary()` function.

```
# merge the effort_Luc.csv and effort_Christina.csv [WARNING: will merge all files named "effort_*" in
directory]
theRefs_screened <- effort_merge()
theRefs_screened[c("STUDY_ID", "REVIEWERS", "INCLUDE")]

##    STUDY_ID REVIEWERS INCLUDE
## 1         2 Christina      NO
## 2         5 Christina      NO
## 3         1       Luc   MAYBE
## 4         3       Luc      NO
## 5         4       Luc     YES
## 6         6       Luc     YES
## 7         7       Luc      NO
## 8         8       Luc   MAYBE
## 9         9       Luc      NO
## 10       10       Luc   MAYBE
## 11       11       Luc   MAYBE

theSummary <- effort_summary(theRefs_screened)

## === SCREENING EFFORT SUMMARY ===
##
##    2 candidate studies identified
##    4 studies excluded
##    5 challenging studies needing additional screening
##   ----
##    11 TOTAL SCREENED
##
## === SCREENING DESIGN SUMMARY ===
##
##           NO MAYBE YES TOTAL         %
## Christina  2     0   0     2  18.18182
## Luc        3     4   2     9  81.81818
## TOTAL      5     4   2    11 100.00000
```

The summary of screening tasks describes the outcomes of which references had studies appropriate for the synthesis project, while also outlining which need to be re-assessed. The team should discuss these challenging references and decide if they are appropriate for inclusion or track down any additional/missing information needed to make proper assessment of their inclusion.

Downloading PDFs

Once references have been screened, **metagear** can be used to download and organize the full-texts of these references. However, note that the download success of these PDFs is entirely conditional on the journal subscription coverage of the host institution running **metagear**. Also note that **metagear** only supports the download of a PDF article if the DOI (digital object identifier) is available for that article.

In this example, we have the following goals:

1. Download a single PDF with the `PDF_download()` function.
2. Download multiple PDFs with the `PDFs_collect()` function.

Let's start by loading the pre-packaged reference dataset in **metagear** that contains the bibliographic information of 11 journal articles (`example_references_metagear`). This dataset includes a column "DOI" which contains the DOI of each article (if available).

```
# load package
library(metagear)
# load a bibliographic dataset with the DOIs
data(example_references_metagear)
# display the year published of each study reference and their DOIs
example_references_metagear[c("JOURNAL", "DOI")]

##                                              JOURNAL                             DOI
## 1  BIOCHEMICAL AND BIOPHYSICAL RESEARCH COMMUNICATIONS    10.1016/j.bbrc.2011.10.017
## 2                         EVOLUTIONARY ECOLOGY RESEARCH                          <NA>
## 3                                  AMERICAN NATURALIST                 10.1086/319928
## 4                                                 GENE    10.1016/j.gene.2008.01.009
## 5                                         VIRUS GENES     10.1007/s11262-012-0864-0
## 6                         JOURNAL OF SHELLFISH RESEARCH          10.2983/035.029.0428
## 7                       JOURNAL OF GENERAL MICROBIOLOGY                          <NA>
## 8                                  APPLIED GEOCHEMISTRY 10.1016/S0883-2927(02)00054-9
## 9      JOURNAL OF DRUG DELIVERY SCIENCE AND TECHNOLOGY                          <NA>
## 10                                   BIOLOGIA PLANTARUM       10.1023/A:1012426306493
## 11                                            GENOMICS   10.1016/j.ygeno.2013.09.002
```

Note that references collected from bibliographic databases like Web of Science will often be incomplete. For example, the study published in EVOLUTIONARY ECOLOGY RESEARCH does not have a DOI (described above as NA). This is because EVOLUTIONARY ECOLOGY RESEARCH is an independently published journal and does not provide DOIs for their research articles.

However, a DOI for the AMERICAN NATURALIST study is available, and let's use it to fetch the PDF.

```
# load package
PDF_download("10.1086/319928", theFileName = "AMNAT_metagear")

## Collecting PDF from DOI: 10.1086/319928
##          Extraction 1 of 2: HTML script.... successful
##          Extraction 2 of 2: PDF download... successful
```

The downloader provides information on the download success, and in this case a PDF was successfully retrieved. It was saved in the working directory of the **R** process (to see this directory use `getwd()`).

Now let's try downloading all the PDFs from our reference dataset. This can be done using the `PDFs_collect()` function.

```
# (optional) initialize the reference dataset to help generate standardized fileNames (e.g., STUDY_ID
numbers)
theRefs <- effort_initialize(example_references_metagear)
# fetch the PDFs
PDFs_collect(theRefs, DOIcolumn = "DOI", FileNamecolumn = "STUDY_ID", directory = getwd())

## Collecting PDF from DOI: 10.1016/j.bbrc.2011.10.017
##          Extraction 1 of 2: HTML script.... successful
##          Extraction 2 of 2: PDF download... successful
## Collecting PDF from DOI: NA
##          Extraction 1 of 2: HTML script.... cannot open: HTTP status was '404 Not Found'
##          Extraction 2 of 2: PDF download... skipped
## Collecting PDF from DOI: 10.1086/319928
##          Extraction 1 of 2: HTML script.... successful
##          Extraction 2 of 2: PDF download... successful
```

```
## Collecting PDF from DOI: 10.1016/j.gene.2008.01.009
##          Extraction 1 of 2: HTML script.... successful
##          Extraction 2 of 2: PDF download... successful
## Collecting PDF from DOI: 10.1007/s11262-012-0864-0
##          Extraction 1 of 2: HTML script.... successful
##          Extraction 2 of 2: PDF download... successful
## Collecting PDF from DOI: 10.2983/035.029.0428
##          Extraction 1 of 2: HTML script.... successful
##          Extraction 2 of 2: PDF download... successful
## Collecting PDF from DOI: NA
##          Extraction 1 of 2: HTML script.... cannot open: HTTP status was '404 Not Found'
##          Extraction 2 of 2: PDF download... skipped
## Collecting PDF from DOI: 10.1016/S0883-2927(02)00054-9
##          Extraction 1 of 2: HTML script.... successful
##          Extraction 2 of 2: PDF download... successful
## Collecting PDF from DOI: NA
##          Extraction 1 of 2: HTML script.... cannot open: HTTP status was '404 Not Found'
##          Extraction 2 of 2: PDF download... skipped
## Collecting PDF from DOI: 10.1023/A:1012426306493
##          Extraction 1 of 2: HTML script.... successful
##          Extraction 2 of 2: PDF download... successful
## Collecting PDF from DOI: 10.1016/j.ygeno.2013.09.002
##          Extraction 1 of 2: HTML script.... successful
##          Extraction 2 of 2: PDF download... successful
##
## PDF download summary
##   8 = downloaded
##   3 = URL error
##  Downloads located in: C:/Users/lajeunesse@usf.edu/Documents
```

Eight of the 11 references had successful PDF downloads; the remaining 3 did not have DOIs available. These PDFs will need to be checked to determine if their contents are the desired research articles. Also note that the downloading process will take time, and in general, it will take ~ 45 seconds to detect and download a single PDF.

Automated extraction of data from scatterplots

Extracting data from a figure image is a common challenge when trying to extract outcomes (effect sizes) from a study. The scrapping (reverse engineering) of data points from a scatterplot image can be automated with **metagear**.

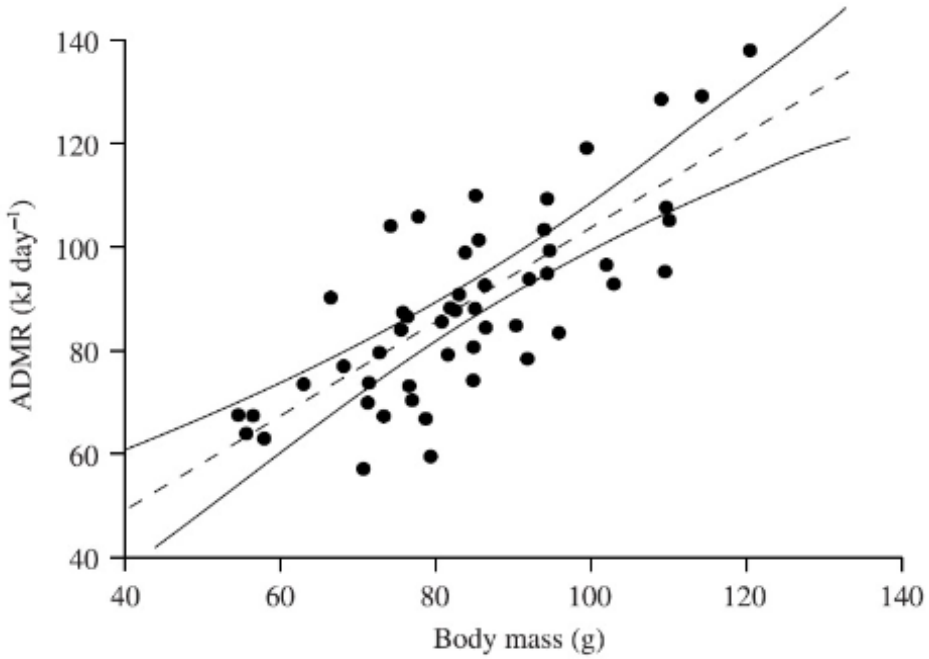
In these examples, we have the following goals:

1. Extract data points from an image containing a scatterplot using the `figure_scatterPlot()` default parameters.
2. Tweak the parameters to extract data from scatterplots with various formats (e.g., different point shapes, or image sizes).

Example 1 | figure_scatterPlot() default settings

Metagear offers a pre-packaged scatterplot image, and so let's begin with extracting data from this image, before moving to more advanced applications of `figure_scatterPlot()`. First, let's load and display the image.

```
# load metagear package and .jpg image manipulation package EBImage
library(metagear)
library(EBImage)
# load the scatterplot image, source: Kam et al. (2003) Functional Ecology 17:496-503.
data(Kam_et_al_2003_Fig2)
# display the image
figure_display(Kam_et_al_2003_Fig2)
```



Now let's use `figure_scatterPlot()` to scrape data from this image; however, because `Kam_et_al_2003_Fig2` is pre-packaged with **metagear** it needs to be converted back to a .jpg before the image can be processed.

The `figure_scatterPlot()` will by default output three objects:

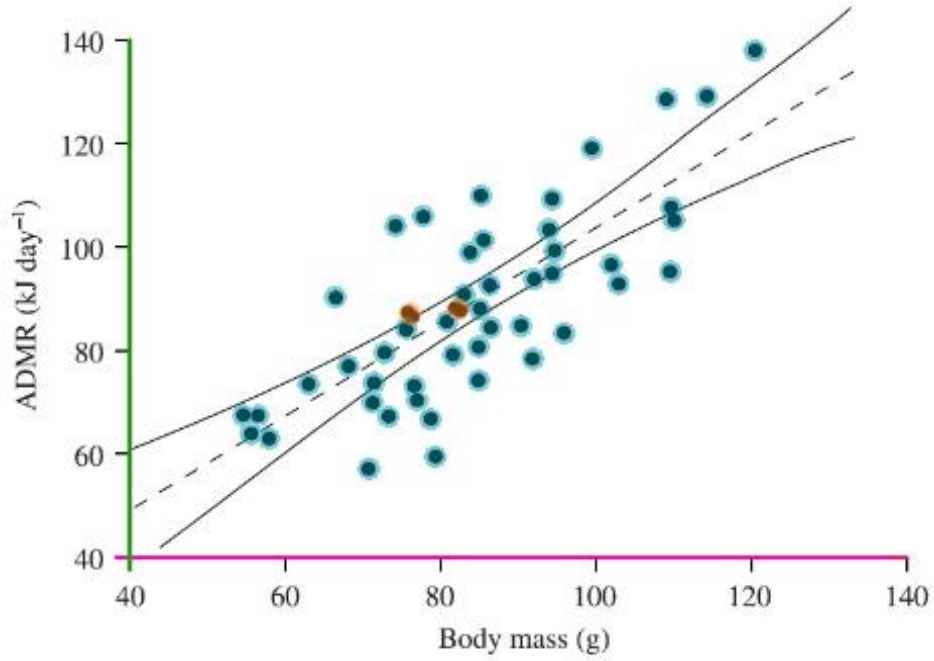
2. The estimated regression fit of these detected points, as well as the estimated effect size and variance of the correlation presented in the figure.
3. A raster image of the detected objects painted over the original image. Blue spheres are detected points, orange spheres are detected clusters of points that could not be separated, the X-axis in pink, and the Y-axis in green. The points and axes can also be extracted individually using the `figure_detectAllPoints()` and `figure_detectAxis()` functions.
4. The X and Y data from each detected point on the image, and information on whether that point was identified as a cluster.

Here are the results of using `figure_scatterPlot()` on Kam et al.'s (2002) figure.

```
# convert back to .jpg
figure_write(Kam_et_al_2003_Fig2, file = "Kam_et_al_2003_Fig2.jpg")
```

```
# load the scatterplot image, source: Kam et al. (2003) Functional Ecology 17:496-503.
rawData <- figure_scatterPlot("Kam_et_al_2003_Fig2.jpg")

## regression fit: Y = 11.92716 + 0.90769 * X, R-squared = 0.59496
## Pearson's r = 0.7713394, var(r) = 0.0034905, N = 49
```

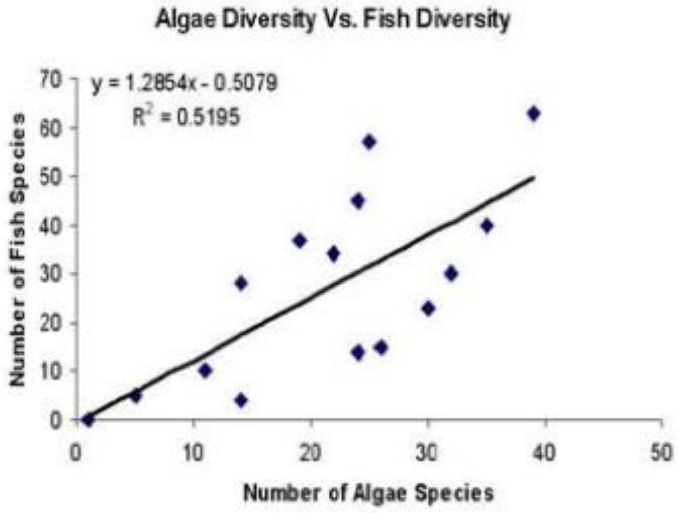


The estimated regression coefficients are very similar to those originally reported by Kam et al.'s (2002) study; which were Y = 12.03 + 0.907 * X with an R2 = 0.59 and a sample size of N = 51.

Example 2 | tweaking defaults for image size

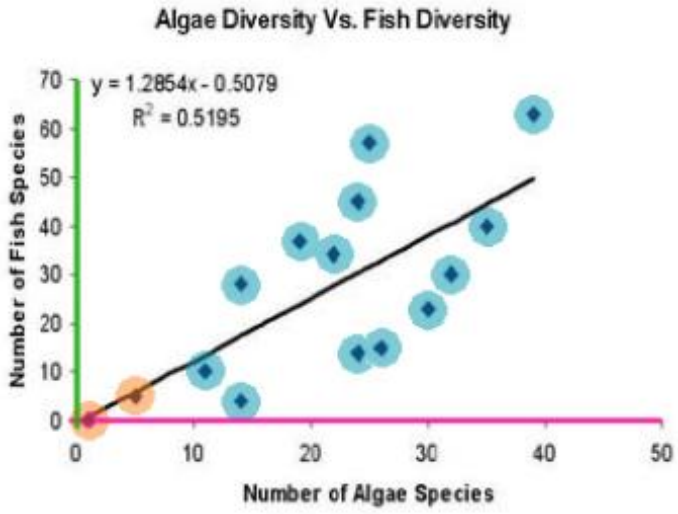
Now let's try to extract data from another image. This time the figure is relatively small and `figure_scatterPlot()` will need some adjustments based on this size difference. Also, this time we will scale the data extractions to the X- and Y-axis scale; this is useful to calculate the original regression coefficients. Here, the minimum and maximum presented in the figure for the X-axis is 0 to 50, and 0 to 70 for the Y-axis. However, note that re-scaling the data does not affect the effect size calculated from the figure, only the estimated regression coefficients. Let's download the image first from my website and then process it.

```
# download the figure image from my website
figureSource <- "http://lajeunesse.myweb.usf.edu/metagear/example_2_scatterPlot.jpg"
download.file(figureSource, "example_2_scatterPlot.jpg", quiet = TRUE, mode = "wb")
aFig <- figure_read("example_2_scatterPlot.jpg", display = TRUE)
```




```
# because of the small size of the image the axis parameter needed adjustment from 5 to 3
rawData2 <- figure_scatterPlot("example_2_scatterPlot.jpg",
                               axis_thickness = 3, # adjusted from 5 to 3 to help detect the thin axis
                               X_min = 0, # minimum X-value reported in the plot
                               X_max = 50, # maximum X-value reported in the plot
                               Y_min = 0,
                               Y_max = 70)

## regression fit: Y = -0.40746 + 1.26962 * X, R-squared = 0.51678
## Pearson's r = 0.7188738, var(r) = 0.0179617, N = 15
```

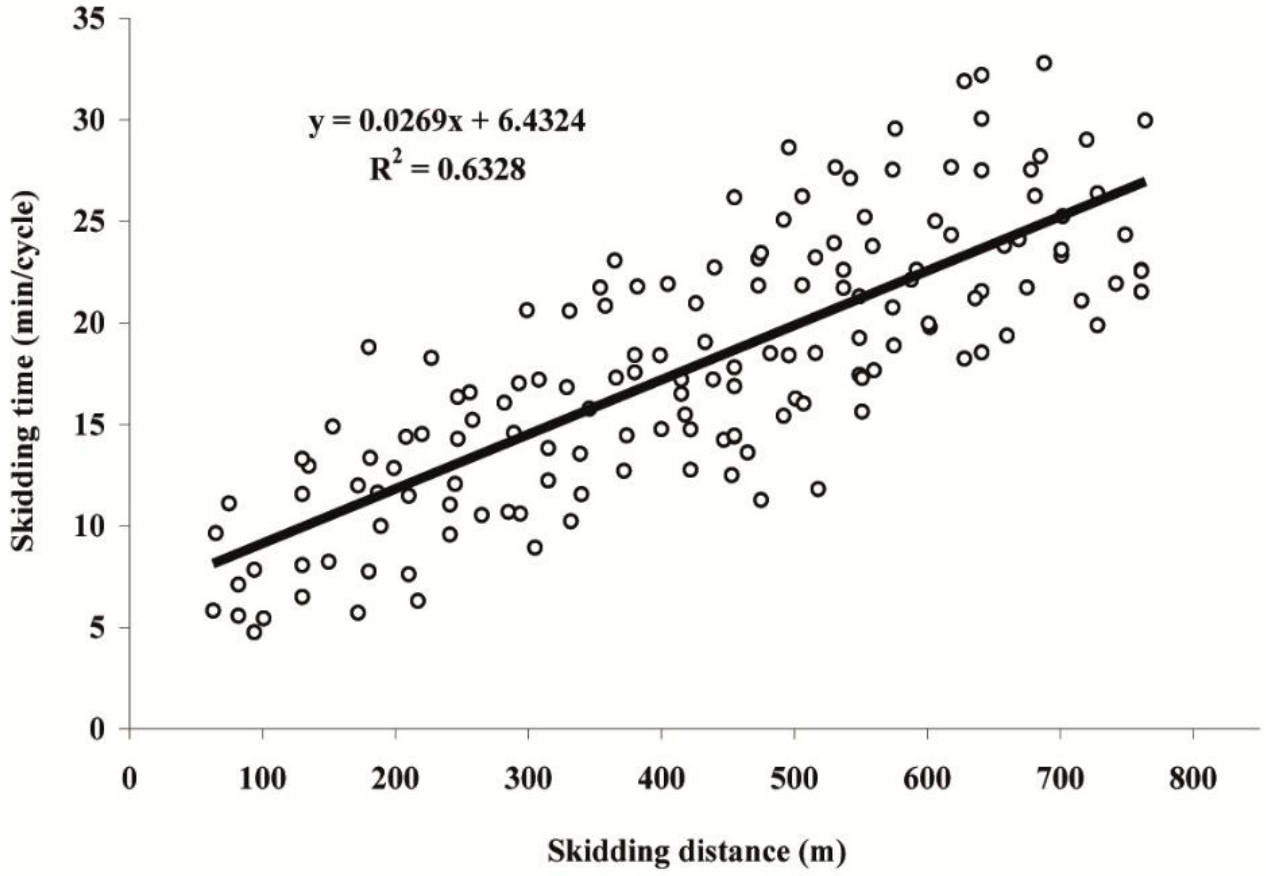


In this example, because of the small size of the figure, the axis_thinkness parameter needed to be reduced from 5 to 3. This was sufficient to detect the axis lines and extract the plotted data.

Example 3 | more tweaking based on color, size, and empty points

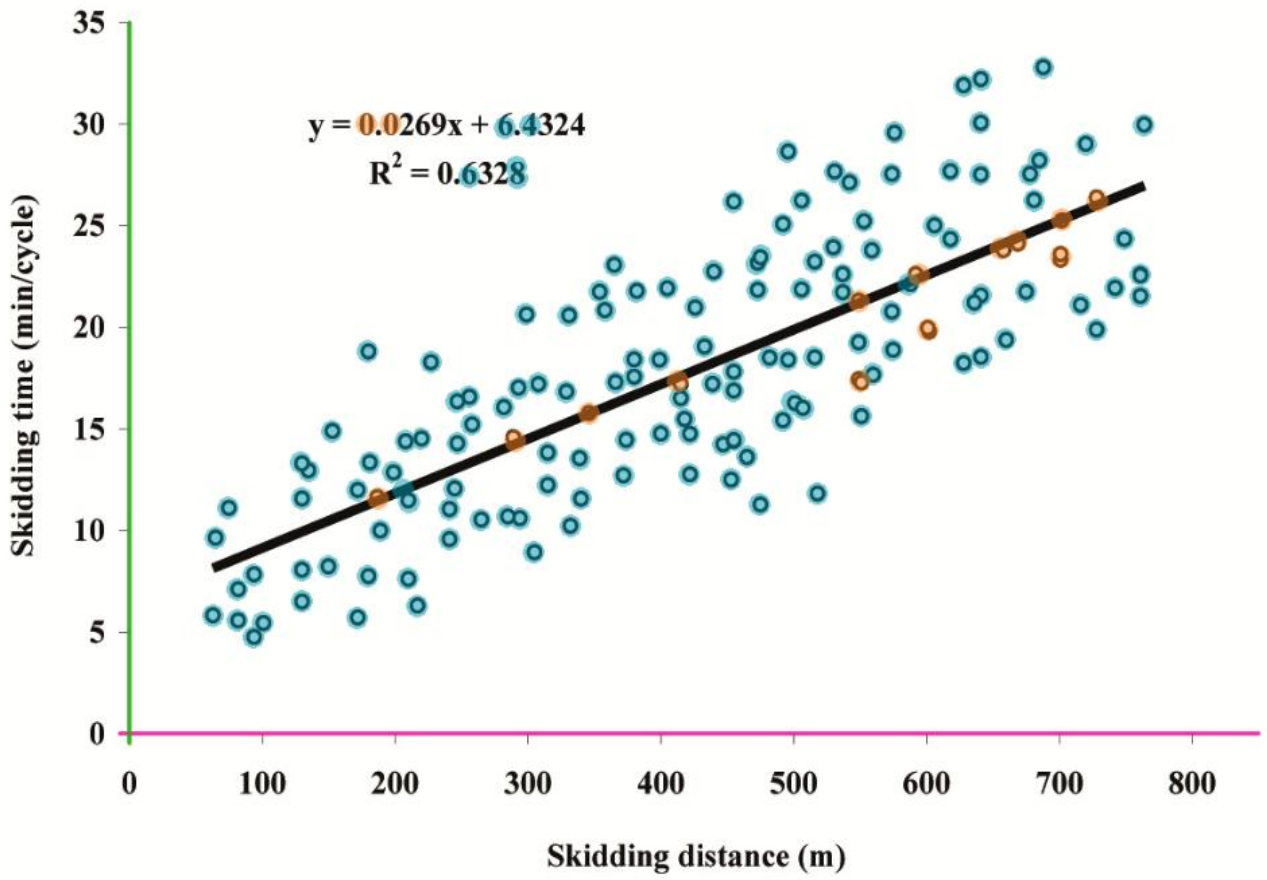
In this figure example, we have the case where the image is large (1122px by 780px), the plotted points are large but empty, and the axis lines are thin and grey. All of these issues complicate object detection on the figure.

```
# download the figure image from my website
figureSource <- "http://lajeunesse.myweb.usf.edu/metagear/example_3_scatterPlot.jpg"
download.file(figureSource, "example_3_scatterPlot.jpg", quiet = TRUE, mode = "wb")
aFig <- figure_read("example_3_scatterPlot.jpg", display = TRUE)
```




```
# tweaking the figure_scatterPlot() function to improve object detection
rawData3 <- figure_scatterPlot("example_3_scatterPlot.jpg",
                               binary_point_fill = TRUE, # set to TRUE to fill empty points
                               point_size = 9, # increase from 5 to 9 since points are large
                               binary_threshold = 0.8, # increase from 0.6 to 0.8 to include the grey
objects
                               axis_thickness = 3, # decrease from 5 to 3 since axes are thin
                               X_min = 0,
                               X_max = 800,
                               Y_min = 0,
                               Y_max = 35)

## regression fit: Y = 8.50394 + 0.02549 * X, R-squared = 0.45299
## Pearson's r = 0.6730416, var(r) = 0.0019557, N = 155
```



It looks like `figure_scatterPlot()` confused some of the regression summary text on the plot for points. This can be avoided by erasing all superfluous information on the figure prior to processing with `figure_scatterPlot()`. However, in our case we are interested in estimating these reported regression coefficients. We can quickly exclude these false detections since they reside within a specific range on the plot that does not include data (e.g., values above 25 for Y, and below 305 for X).

```
# remove false detected points from the regression summary presented within the plot
cleaned_rawData3 <- rawData3[ which(!(rawData3$X < 350 & rawData3$Y > 25)), ]
# estimate the regression coefficients
lm(Y ~ X, data = cleaned_rawData3)

##
## Call:
## lm(formula = Y ~ X, data = cleaned_rawData3)
##
## Coefficients:
## (Intercept)            X
##     6.45334      0.02893

# and get R-squared
round(summary(lm(Y ~ X, data = cleaned_rawData3))$r.squared, 4)

## [1] 0.6369
```

The estimated regression coefficients are very similar to those presented within the plot.

Automated extraction of data from bar plots

Bar plots (or bar charts) are a common way to present information in groups or categories.

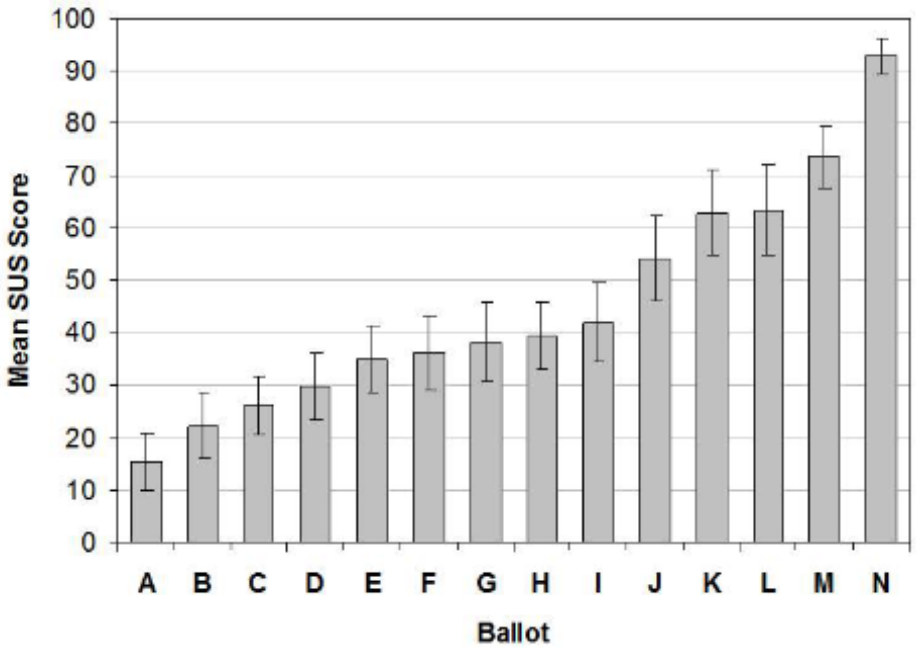
In these examples, we have the following goals:

1. Extract data points from an image containing a bar plot using the `figure_barPlot()` default parameters.
2. Tweak the parameters to extract data from bar plots with various formats (e.g., with bars with different shading indicating different groups, or bars presented horizontally rather than vertically).

Example 1 | figure_barPlot() default settings

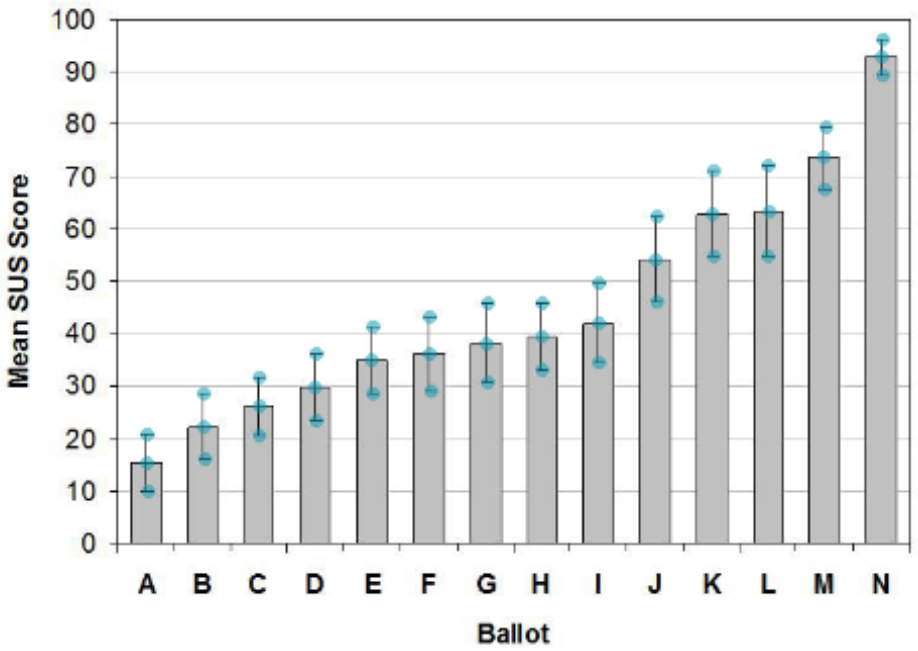
Let's have a look at the bar plot image provided by **metagear** called `Kortum_and_Acymyan_2013_Fig4`; originally extracted from Kortum & Acymyan (2013; Journal of Usability Studies 9:14-24).

```
# load metagear package
library(metagear)
# load the scatterplot image, source: Kortum & Acymyan (2013) J. of Usability Studies 9:14-24).
data(Kortum_and_Acymyan_2013_Fig4)
# display the image
figure_display(Kortum_and_Acymyan_2013_Fig4)
```



Manual extraction of the bars and their errors will be time consuming here given that there are 42 separate data points to be gathered (i.e. 14 bars each with upper and lower error bars). Let's use `figure_barPlot()` with its default options to extract these 42 points.

```
# convert metagear image object back to .jpg and then extract objects from this .jpg
figure_write(Kortum_and_Acymyan_2013_Fig4, file = "Kortum_and_Acymyan_2013_Fig4.jpg")
rawData <- figure_barPlot("Kortum_and_Acymyan_2013_Fig4.jpg")
```



In the above image, the detected points for each ballot were painted in blue. Let's have a closer look at these extracted data.

```
# display extracted points
as.vector(round(rawData, 2))
```

```
##  [1] 15.13 20.57  9.69 21.99 28.37 15.84 31.44 26.00 20.33 29.55 35.93 23.17 34.75 28.37 41.13 35.93
43.03 28.84 37.83 45.63 30.50 32.86 39.24 45.63 41.84 34.28 49.41 53.90 62.17 45.86 62.65 54.61 70.92
71.87 54.61 63.12 73.52 67.38 79.20 92.67 89.13 95.98
```

Metagear is not clever enough to know what groupings these extractions belong too; however, the extractions will be sorted relative to their axis positioning. For example, there are three extractions that occupy the same X-axis range under the A ballot column. These three extractions will be grouped together in the `figure_barPlot()` output. With this in mind, a little data manipulation is needed to make better sense of these ballot data.

```
# extractions are in triplicates with an upper, mean, and lower values, so let's
# stack by three and sort within triplicates from lowest to highest
organizedData <- t(apply(matrix(rawData, ncol = 3, byrow = TRUE), 1, sort))
# rename rows and columns of these triplicates as presented in Kortum_and_Acymyan_2013_Fig4.jpg
theExtraction_names <- c("lower 95%CI", "mean SUS score", "upper 95%CI")
theBar_names <- toupper(letters[1:14])
dimnames(organizedData) <- list(theBar_names, theExtraction_names)
organizedData
```

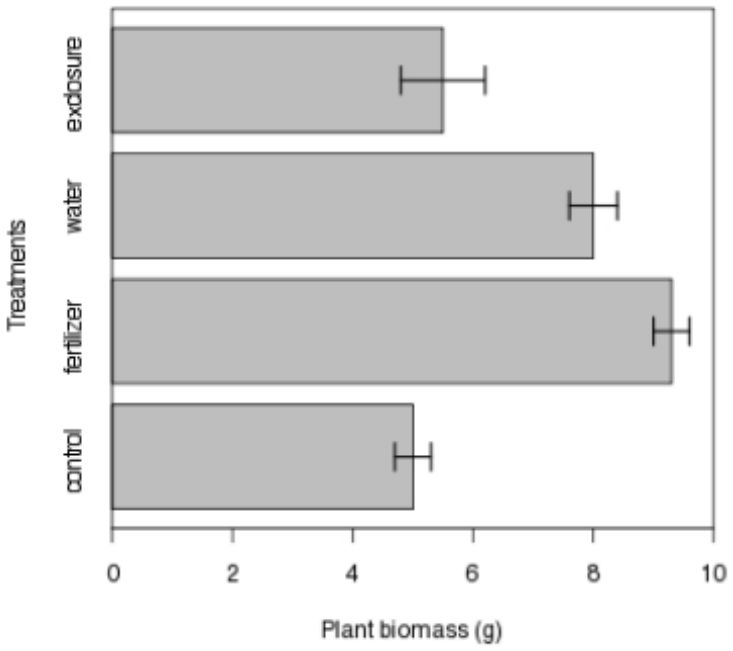
```
##   lower 95%CI mean SUS score upper 95%CI
## A    9.692671       15.13002    20.56738
## B   15.839243       21.98582    28.36879
## C   20.330969       26.00473    31.44208
```

```
## D   23.167849       29.55083    35.93381
## E   28.368794       34.75177    41.13475
## F   28.841608       35.93381    43.02600
## G   30.496454       37.82506    45.62648
## H   32.860520       39.24350    45.62648
## I   34.278960       41.84397    49.40898
## J   45.862884       53.90071    62.17494
## K   54.609929       62.64775    70.92199
## L   54.609929       63.12057    71.86761
## M   67.375887       73.52246    79.19622
## N   89.125296       92.67139    95.98109
```

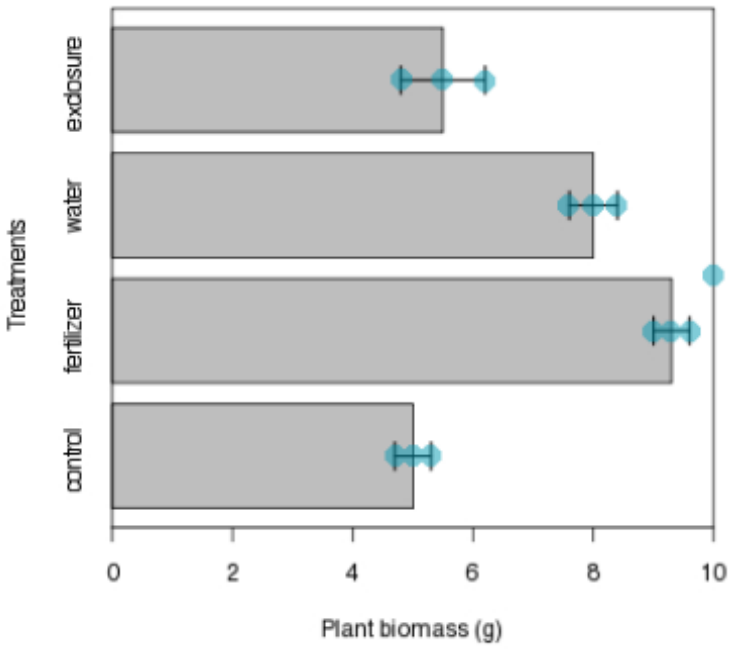
Example 2 | tweaking defaults for horizontal columns

Now let's try to extract data from another image where bar-plot is presented horizontally (i.e. bars stem from the Y-axis).

```
# download the figure image from my website
figureSource <- "http://lajeunesse.myweb.usf.edu/metagear/example_2_barPlot.jpg"
download.file(figureSource, "example_2_barPlot.jpg", quiet = TRUE, mode = "wb")
aFig <- figure_read("example_2_barPlot.jpg", display = TRUE)
```




```
rawData2 <- figure_barPlot("example_2_barPlot.jpg",
                           horizontal = TRUE, # changed from FALSE since bars are horizontal
                           bar_width = 11, # raised from 9 since bars are wide relative to the figure
                           Y_min = 0,
                           Y_max = 10)
```



The function also detected the right-most vertical line (part of the the figure box) as a datapoint. The options of `figure_barPlot()` can be tweaked to avoid this issue; however, it might be easier to just exclude this extraction given that it has the largest plant biomass value (i.e. close to 10). Let's exclude this false datapoint and organize the dataset as presented in the figure.

```
# exclude the false detection
rawData2 <- rawData2[rawData2 < max(rawData2)]
# data are in triplicates with an upper, mean, and lower values, so let's
# stack by three and sort within triplicates from lowest to highest
organizedData <- t(apply(matrix(rawData2, ncol = 3, byrow = TRUE), 1, sort))
# rename rows and columns of these triplicates as presented in the figure
theExtraction_names <- c("lower error", "bar", "upper error")
theBar_names <- c("exclosure", "water", "fertilizer", "control")
dimnames(organizedData) <- list(theBar_names, theExtraction_names)
organizedData
```

```
##            lower error      bar upper error
## exclosure     4.775438 5.466238    6.173633
## water         7.572347 7.974277    8.376206
## fertilizer    8.986066 9.276230    9.581994
## control       4.678975 4.983923    5.273312
```

Meta-analysis with multiple effect sizes that share a common control

Typically an effect size quantified with a response ratio uses the means (X), standard deviations (SD), and sample sizes (N) from single control (C) and treatment (T) groups. However, some studies will compare multiple treatment groups to a single control.

Here we will replicate the meta-analysis example presented in Lajeunesse (2011; Ecology 92, 2049-2055) for modeling effect sizes that share a common control.

```
# load metagear package
library(metagear)
# get dataset from my website
dataSource <- "http://lajeunesse.myweb.usf.edu/metagear/Lajeunesse_2011_commonControl.csv"
theData <- read.csv(dataSource, header = TRUE)
# calculate response ratios (RR) and add these effect sizes to the dataset
theData$RR <- log(theData$X_T/theData$X_C)
# display effect sizes as reported by Lajeunesse (2011; page 2052, second paragraph)
round(theData$RR, 3)

## [1] -0.598  0.182  0.718
```

These three RR effect sizes share a common control. The next step is to model the covariances (the dependencies) among these effect sizes using the **metagear**'s `covariance_commonControl()` function. There will be a list of two objects outputted from this function, the first will be the variance-covariance matrix that models the dependencies among effect sizes, and the second is the effect size dataset that is aligned with the structure of this matrix. Let's now compute and display the matrix.

```
# estimate the sample variance-covariance (VCV) matrix that models the common control relationships among
RR
V <- covariance_commonControl(theData, "commonControl_ID", "X_T", "SD_T", "N_T", "X_C", "SD_C", "N_C",
metric = "RR")
# display the VCV matrix with rounded variances and covariances
round(V[[1]], 3)

##       [,1]  [,2]  [,3]
## [1,] 0.105 0.047 0.047
## [2,] 0.047 0.087 0.047
## [3,] 0.047 0.047 0.060
```

Note the off-diagonals of the matrix are non-zero; this structure models the shared variance (covariance) among the three effect sizes due to the common control. The equation for the common-control covariance is simple: (X_C ^ 2) / N_C .

Now let's use this matrix to model the dependent effect sizes in a meta-analysis. Here we will conduct a simple fixed-effect meta-analysis as presented by Lajeunesse (2011) using the **metafor** R package.

```
# perform a random-effects meta-analysis on these effect sizes using the metafor R package
suppressWarnings(suppressMessages(library(metafor))) # remove all messages when loading package
theCovarianceMatrix <- V[[1]]
theAlignedData <- V[[2]]
rma.mv(RR, # a simple model that only pools the 3 effect sizes
       V = theCovarianceMatrix, # inclusion of the sample VCV matrix
       data = theAlignedData, # the dataset with the effect sizes
       method = "FE", # "FE" = fixed effect
       digits = 4)
```

```
##
## Multivariate Meta-Analysis Model (k = 3; method: FE)
##
## Variance Components: none
##
## Test for Heterogeneity:
## Q(df = 2) = 25.8185, p-val < .0001
##
## Model Results:
##
## estimate       se     zval     pval    ci.lb    ci.ub
##   0.4054   0.2356   1.7207   0.0853  -0.0564   0.8671        .
##
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The pooled effect size sharing a common control was 0.41 with a variance of 0.0556 (converting SE to variance with 0.2356 ^ 2).