

bmbstats: bootstrap magnitude-based statistics for sports scientists

Mladen Jovanovic

2020-07-29

Contents

Welcome	9
R and R packages	9
License	10
I Part One	11
1 Introduction	13
2 Description	17
2.1 Comparing two independent groups	17
2.2 Comparing dependent groups	30
2.3 Describing relationship between two variables	36
2.4 Advanced uses	46
3 Prediction	47
3.1 Overfitting	48
3.2 Cross-Validation	52
3.3 Bias-Variance decomposition and trade-off	58
3.4 Interpretability	63
3.5 Magnitude-based prediction estimators	64
3.6 Practical example: MAS and YoYoIR1 prediction	66

4 Causal inference	77
4.1 Necessary versus sufficient causality	77
4.2 Observational data	78
4.3 Potential outcomes or counterfactuals	80
4.4 <i>Ceteris paribus</i> and the biases	82
4.5 Subject matter knowledge	85
4.6 Example of randomized control trial	86
4.7 Prediction as a complement to causal inference	94
4.8 Ergodicity	121
5 Statistical inference	125
5.1 Two kinds of uncertainty, two kinds of probability, two kinds of statistical inference	126
6 Frequentist perspective	129
6.1 Null-Hypothesis Significance Testing	132
6.2 Statistical Power	136
6.3 New Statistics: Confidence Intervals and Estimation	138
6.4 Minimum Effect Tests	141
6.5 Magnitude Based Inference	146
7 Bayesian perspective	149
7.1 Grid approximation	149
7.2 Priors	150
7.3 Likelihood function	151
7.4 Posterior probability	153
7.5 Adding more possibilities	156
7.6 Different prior	157
7.7 More data	158
7.8 Summarizing prior and posterior distributions with MAP and HDI	159
7.9 Comparison to NHST Type I errors	160

CONTENTS	5
8 Bootstrap	163
8.1 Summarizing bootstrap distribution	165
8.2 Bootstrap Type I errors	166
9 Statistical inference conclusion	169
10 Measurement Error	171
10.1 Estimating TE using <i>ordinary least products</i> regression	177
10.2 Smallest Detectable Change	179
10.3 Interpreting individual changes using SESOI and SDC	180
10.4 What to do when we know the error?	184
10.5 Extending the Classical Test Theory	188
11 Conclusion	191
II Part Two	193
12 bmbstats: Bootstrap Magnitude-based Statistics package	195
12.1 bmbstats Installation	195
13 Descriptive tasks using bmbstats	197
13.1 Generating <i>height data</i>	197
13.2 Visualization and analysis of a single group/variable	198
13.3 Visualization and analysis of the two independent groups	206
13.4 NHST, METs and MBI functions	213
13.5 Comparing two dependent groups	219
13.6 Describing relationship between two groups	242
14 Predictive tasks using bmbstats	247
14.1 How to implement different performance metrics?	251
14.2 How to use different prediction model?	253
14.3 Example of using tuning parameter	255
14.4 Plotting	258

14.5 Comparing models	263
14.6 Bootstrapping model	271
15 Validity and Reliability	275
15.1 Data generation	275
15.2 Validity	276
15.3 Reliability	304
15.4 Repeatability	316
15.5 The difference between Reproducibility and Repeatability	321
16 RCT analysis and prediction in <code>bmbstats</code>	327
16.1 Data Generating Process behind RCT	327
16.2 RCT analysis using <code>bmbstats::RCT_analysis</code> function	332
16.3 Linear Regression Perspective	344
16.4 Prediction perspective 1	351
16.5 Adding some effects	360
16.6 What goes inside the <i>measurement error</i> (or Control group change or residuals SD)?	370
16.7 Prediction perspective 2	373
16.8 Making it more complex by adding covariate	378
16.9 Prediction perspective 3	389
17 Appendix A: <code>dorem</code> package	411
17.1 <code>dorem</code> Installation	411
17.2 <code>dorem</code> Example	411
18 Appendix B: <code>shorts</code> package	417
18.1 <code>shorts</code> Installation	417
18.2 <code>short</code> Examples	417
18.3 <code>shorts</code> Citation	438

CONTENTS	7
19 Appendix C: vjsim package	439
19.1 vjsim Installation	439
19.2 vjsim Usage	439
19.3 Shiny App ¹	441
19.4 vjsim Example	441
20 Appendix D: Recommended material	449
21 References	453

¹<https://athletess.shinyapps.io/shiny-simulator/>

Welcome

The aim of this book is to provide an overview of the three classes of tasks in the statistical modeling: description, prediction and causal inference (Hernán, Hsu, and Healy 2019). Statistical inference is often required for all three tasks. Short introduction to frequentist null-hypothesis testing, Bayesian estimation and bootstrap are provided. Special attention is given to the practical significance with the introduction of magnitude-based estimators and statistical inference by using the concept of smallest effect size of interest (SESOI). Measurement error is discussed with the particular aim of interpreting individual change scores. In the second part of this book, common sport science problems are introduced and analyzed with the **bmbstats** package.

This book, as well as the **bmbstats** package are in active opens-source development. Please be free to contribute pull request at GitHub when you spot an issue or have an improvement idea. I am hoping both this book and the **bmbstats** package to be collaborative tools that can help both up-and-coming as well as experienced researchers and sports scientists.

bmbstats package

<https://github.com/mladenjovanovic/bmbstats>

bmbstats book

<https://github.com/mladenjovanovic/bmbstats-book>

R and R packages

This book is fully reproducible and was written in R (Version 4.0.0; R Core Team 2020) and the R-packages *automatic* (Lang et al. 2014), *bayestestR* (Version 0.7.2; Makowski, Ben-Shachar, and Lüdecke 2019), *bmbstats* (Version 0.0.0.9001; Jovanović 2020a), *bookdown* (Version 0.20.2; Xie 2016), *boot* (Version 1.3.25; A.

C. Davison and Hinkley 1997a), *carData* (Version 3.0.4; Fox, Weisberg, and Price 2019), *caret* (Version 6.0.86; Kuhn 2020), *cowplot* (Version 1.0.0; Wilke 2019), *directlabels* (Version 2020.6.17; Hocking 2020), *dorem* (Version 0.0.0.9000; Jovanovic and Hemingway 2020), *dplyr* (Version 1.0.0; Wickham, François, et al. 2020), *effects* (Version 4.1.4; Fox and Weisberg 2018; Fox 2003; Fox and Hong 2009), *forcats* (Version 0.5.0; Wickham 2020), *ggplot2* (Version 3.3.2; Wickham 2016), *ggridges* (Version 0.5.2; Wilke 2020), *ggstance* (Version 0.3.4; Henry, Wickham, and Chang 2020), *hardhat* (Version 0.1.4; Vaughan and Kuhn 2020), *kableExtra* (Version 1.1.0; Zhu 2019), *knitr* (Version 1.29; Xie 2015), *lattice* (Version 0.20.41; Sarkar 2008), *markdown* (Version 1.1; Allaire et al. 2019), *Metrics* (Version 0.1.4; Hamner and Frasco 2018), *minerva* (Version 1.5.8; Albanese et al. 2012a), *mlr* (Version 2.17.1; Bischl et al. 2016, 2017; Lang et al. 2019), *mlr3* (Version 0.4.0; Lang et al. 2019), *mlrmbo* (Bischl et al. 2017), *multilabel* (Probst et al. 2017), *nlme* (Version 3.1.148; Pinheiro et al. 2020), *openml* (Casalicchio et al. 2017), *ParamHelpers* (Version 1.14; Bischl et al. 2020), *pdp* (Version 0.7.0; B. M. Greenwell 2017a), *psych* (Version 1.9.12.31; Revelle 2019), *purrr* (Version 0.3.4; Henry and Wickham 2020), *readr* (Version 1.3.1; Wickham, Hester, and Francois 2018), *rpart* (Version 4.1.15; Therneau and Atkinson 2019), *shorts* (Version 1.1.0; Jovanovic 2020), *stringr* (Version 1.4.0; Wickham 2019), *tibble* (Version 3.0.3; Müller and Wickham 2020), *tidyverse* (Version 1.3.0; Wickham, Averick, et al. 2019), *vip* (Version 0.2.2; Greenwell, Boehmke, and Gray 2020), *visreg* (Version 2.7.0; Breheny and Burchett 2017), and *vjsim* (Version 0.1.1.9000; Jovanović 2020b).

License

This work, as a whole, is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License.

The code contained in this book is simultaneously available under the MIT license²; this means that you are free to use it in your own packages, as long as you cite the source.

²<https://opensource.org/licenses/MIT>

Part I

Part One

Chapter 1

Introduction

The *real* world is very complex and uncertain. In order to help in understanding it and to predict its behavior, we create maps and models (Page 2018; Weinberg and McCann 2019). One such tool are statistical models, representing a simplification of the complex and ultimately uncertain *reality*, in the hope of describing it, understanding it, predicting its behavior, and help in making decisions and interventions (Hernán, Hsu, and Healy 2019; Lang, Sweet, and Grandfield 2017; McElreath 2015; Pearl and Mackenzie 2018). In the outstanding statistics book “Statistical Rethinking” (McElreath 2015, 19), the author stresses the distinction between *Large World* and *Small World*, described initially by Leonard Savage (Binmore 2011; Gigerenzer, Hertwig, and Pachur 2015; Savage 1972):

"All statistical modeling has these same two frames: the small world of the model itself and the large world we hope to deploy the model in. Navigating between these two worlds remains a central challenge of statistical modeling. The challenge is aggravated by forgetting the distinction.

The small world is the self-contained logical world of the model. Within the small world, all possibilities are nominated. There are no pure surprises, like the existence of a huge continent between Europe and Asia. Within the small world of the model, it is important to be able to verify the model’s logic, making sure that it performs as expected under favorable assumptions. Bayesian models have some advantages in this regard, as they have reasonable claims to optimality: No alternative model could make better use of the information in the data and support better decisions, assuming the small world is an accurate description of the real world.

The large world is the broader context in which one deploys a model. In the large world, there may be events that were not imagined

in the small world. Moreover, the model is always an incomplete representation of the large world, and so will make mistakes, even if all kinds of events have been properly nominated. The logical consistency of a model in the small world is no guarantee that it will be optimal in the large world. But it is certainly a warm comfort.”

Creating “Small Worlds” relies heavily on making and accepting numerous assumptions, both known and unknown, as well as *prior* expert knowledge, which is ultimately incomplete and fallible. Because all statistical models require subjective choices (Gelman and Hennig 2017), there is no *objective* approach to make “Large World” inferences. It means that it must be us who make the inference, and claims about the “Large World” will always be uncertain. Additionally, we should treat statistical models and statistical results as being much more incomplete and uncertain than the current norm (Amrhein, Trafimow, and Greenland 2019).

We must accept the *pluralism* of statistical models and models in general (Mitchell 2012, 2002), move beyond subjective-objective dichotomy by replacing it with virtues such as *transparency*, *consensus*, *impartiality*, *correspondence to observable reality*, *awareness of multiple perspectives*, *awareness of context-dependence*, and *investigation of stability* (Gelman and Hennig 2017). Finally, we need to accept that we must act based on *cumulative knowledge* rather than solely rely on single studies or even single lines of research (Amrhein, Trafimow, and Greenland 2019).

This discussion is the topic of epistemology, scientific inference, and philosophy of science, thus far beyond the scope of the present book (and the author). Nonetheless, it was essential to convey that statistical modeling is a process of creating the “Small Worlds” and deploying it in the “Large World”. There are three main classes of tasks that the statistical model is hoping to achieve: *description*, *prediction*, and *causal inference* (Hernán, Hsu, and Healy 2019).

The following example will help in differentiating between these three classes of tasks. Consider a king who is facing a drought who must decide whether to invest resources in rain dances. The queen, upon seeing some rain clouds in the sky, must decide on whether to carry her umbrella or not. Young prince, who likes to gamble during his hunting sessions, is interested in knowing what region of his father’s vast Kingdom receives the most rain. All three would benefit from an empirical study of rain, but they have different requirements of the statistical model. The king requires *causality*: Do rain dances cause rain? The queen requires *prediction*: Does it look likely enough to rain for me to ask my servants to get my umbrella? The prince requires simple quantitative summary *description*: have I put my bets on the correct region?

The following sections will provide an overview of the three classes of tasks in the statistical modeling. Data can be classified as being on one of four scales: *nominal*, *ordinal*, *interval* or *ratio* and description, prediction and causal techniques differ depending on the scales utilized. For the sake of simplicity and

big picture overview, only examples using ratio scale are to be considered in this book.

Chapter 2

Description

Description provides *quantitative summary* of the acquired data sample. These quantitative summaries are termed *descriptive statistics* or *descriptive estimators* and are usually broken down into two main categories: *measures of central tendency*, and *measures of spread / dispersion*. The stance taken in this book is that descriptive statistics involve *all* quantitative summaries (or *aggregates*) that are used to describe data without making predictive or causal claims. For example, *linear regression* between two variables can be used as a descriptive tool if the aim is to measure *linear association* between two variables, but it can be also used in predictive and causal tasks. *Effect sizes* such as *change*, *percent change* or *Cohen's d* represent descriptive statistics used to compare two or more groups, and are commonly used in causal tasks to estimate *average causal effect* of the treatment.

To provide further explanation of the descriptive statistics, three common descriptive tasks in sport science are given as examples: (1) comparing two independent groups, (2) comparing two dependent groups, (3) measuring association between two variables.

2.1 Comparing two independent groups

Imagine we carried collection of body height measurements and we obtained N=100 observations using N=50 female and N=50 male subjects. Collected data is visualized in Figure 2.1.

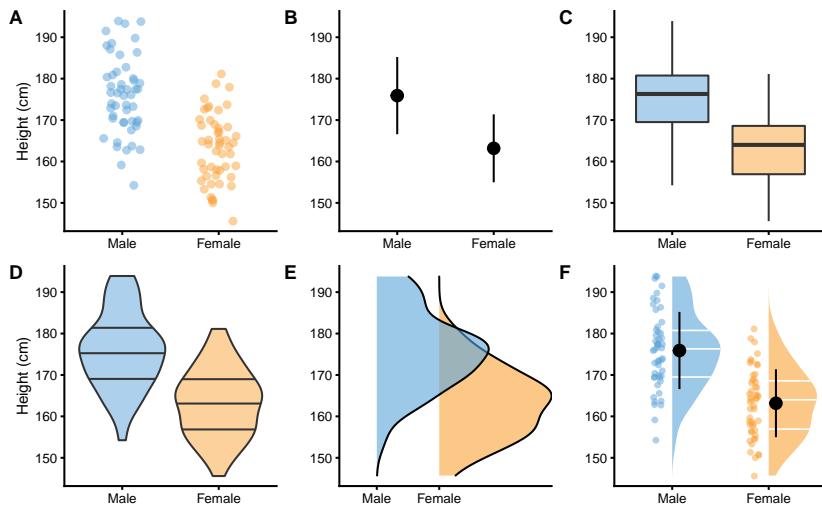


Figure 2.1: Common techniques to visualize independent groups observations. Before any analysis takes place, it is always a good practice to visualize the data first. Ideally, we want to visualize the complete data set, rather than only provide descriptive summaries, such as means. **A.** Simple scatter-plot with jitter to avoid overlap between the points. **B.** Mean and standard deviation as error bars. **C.** Box-plot. Horizontal line represents median, or 50th percentile, whereas boxes represent 25th and 75th percentile. Vertical lines usually represent min and max, although they can extend up to 1.5xIQR (inter-quartile range) with point outside of that interval plotted as *outliers*. **D.** Violin plots representing double-side density plots with 25th, 50th and 75th percentile lines. **E.** Density plots indicating sample distribution. **F.** Raincloud plot (Allen et al. 2019, 2018) which combine kernel density plots as *clouds* with accompanying 25th, 50th and 75th percentile lines, mean \pm SD error bars and jittered points as *rain*

Commonly provided descriptive statistics for each group can be found in the Table 2.1. **Mean**, **median** and **mode** are common measures of central tendencies. **Standard deviation (SD)**, **median absolute difference (MAD)**, **inter-quartile range (IQR)**, **min**, **max** and **range** are common measures of spread or dispersion. **Percent coefficient of variation (% CV)** is also a measure of dispersion, but **standardized**¹ which allows comparison of variables that are on different scales. **Skewness (skew)** is usually described as a measure of a symmetry. A perfectly symmetrical data set will have a skewness of 0. **Kurtosis** measures the tail-heaviness of the distribution. More in depth discussion of descriptive estimators, particularly

¹ Standardization is the process of putting different variables on the same scale. This allows for easier comparison, as well as graphing using a common axis. For example, variables are usually standardized by using Z-Score ($z_i = \frac{x_i - \bar{x}}{SD_x}$) which has a mean of zero and a standard deviation of 1.

Table 2.1: Common descriptive statistics or estimators

Estimator	Male	Female
n	50.00	50.00
mean (cm)	175.90	163.18
SD (cm)	9.32	8.20
% CV	5.30	5.02
median (cm)	176.30	164.00
MAD (cm)	9.52	8.86
IQR (cm)	11.24	11.67
mode (cm)	176.26	164.94
min (cm)	154.24	145.59
max (cm)	193.90	181.12
range (cm)	39.66	35.53
skew	0.08	0.08
kurtosis	-0.53	-0.69

robust estimators (Rousselet, Pernet, and Wilcox 2017; Wilcox, Peterson, and McNitt-Gray 2018; Wilcox and Rousselet 2017; Wilcox 2016) is beyond the topic of this short overview.

2.1.1 Sample mean as the simplest statistical model

In the [Introduction](#) of this book, statistical models are defined as “Small Worlds” or simplifications of the complex and uncertain reality. From this perspective, sample [mean](#) can be considered the simplest statistical model. With this estimator we are representing all of the data points with one quantitative summary (i.e. *aggregate*). However, how do we choose an estimate that represents the sample the best? Estimate that has the minimal *error* is selected as the *optimal* representative. Error is defined using a *loss function* that penalizes difference between the model estimate or prediction (\hat{y}_i) and observations (y_i) (Equation (2.1)). The difference between model prediction (\hat{y}_i) and observations (y_i) is called *residual*.

$$\text{Loss function} = f(\text{observed}, \text{predicted}) \quad (2.1)$$

Two most common loss functions are *absolute loss* (also referred to as *L1*) (Equation (2.2)) and *quadratic loss* (also referred to as *squared errors* or *L2*) (Equation (2.3)). Please refer to section [Sample mean as the simplest predictive model](#) in [Prediction](#) chapter for more examples.

$$\text{absolute loss} = |\hat{y}_i - y_i| \quad (2.2)$$

$$\text{quadratic loss} = (\hat{y}_i - y_i)^2 \quad (2.3)$$

Cost function is an *aggregate* of the loss function (Equation (2.4)).

$$\text{Cost function} = f(\text{Loss function}(observed, predicted)) \quad (2.4)$$

Since loss function is defined on a data point (i.e. y_i), we need to aggregate losses into a single metric. This is done with a cost function, usually using `sum` or `mean`.

One such cost function is *root-mean-square-error* (**RMSE**) (Equation (2.5)). **RMSE** takes the square root of the mean of the quadratic loss (note the $(\hat{y}_i - y_i)^2$ in the **RMSE** equation, which represent quadratic loss). **RMSE** thus represents a measure of the *model fit*, or how good the model fits the data. Lower **RMSE** means lower error and thus a better fit.

$$RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^n (\hat{y}_i - y_i)^2} \quad (2.5)$$

By using body height data from the female group, we can *search* for a body height estimate that minimizes the **RMSE** (Figure 2.2). That body height estimate would be considered the best representative of the sample, and thus the simplest statistical model.

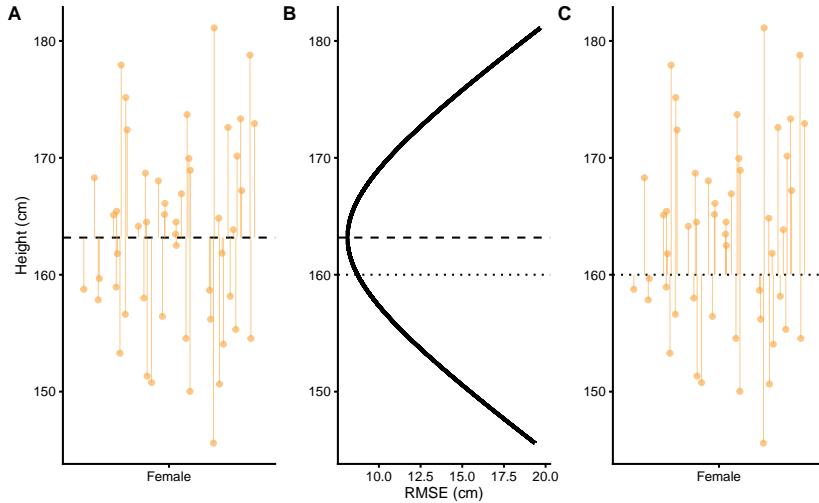


Figure 2.2: Sample mean as the simplest statistical model. **A.** Dashed line represents the estimate, in this case the `mean` of the sample. Vertical line represent residuals between estimate and observed values. **B.** Each estimate has a `RMSE` value. Central tendency estimate with the lowest `RMSE` value is the sample `mean`. **C.** Similar to panel A, this panel depicts residuals for a central tendency estimate with higher `RMSE`

As the result of this search, the body height estimate that minimizes the error is 163.18cm, and accompanying RMSE is equal to 8.12cm. As it can be read from the Table 2.1, this optimal body height estimate is equal to calculated sample `mean`. Standard deviation of the sample is equal to `RMSE`². From statistical modeling perspective, sample mean can be considered sample estimate that minimizes the sample `SD`, and sample `SD` can be seen as the measure of the model fit.

This search for the optimal estimate that minimizes the cost function can be expanded to other statistical models. For example, linear regression can be seen as a search for the line that minimizes `RMSE`. This approach of estimating model parameters or estimators belongs to the family of *ordinary least squares* (OLS) methods, although there are other approaches such as *maximum likelihood estimation* (MLE) which will be discussed in [Statistical inference](#) section (Foreman 2014). The solutions to some of these models can be found *analytically*³, but for

²As can be noticed, `RMSE` and `SD` are not exactly the same. This is because a sample `SD` equation uses $n - 1$ instead of n : $SD = \sqrt{\frac{1}{n-1} \sum_{i=1}^n (y_i - \bar{y})^2}$, where \bar{y} represents the `mean`. Remember that \hat{y}_i represents the model estimate. In this case model estimate \hat{y}_i and sample `mean` \bar{y} are the same. Sample `SD` uses $n - 1$ since this represents *unbiased estimator* of the *population SD*. More about this topic will be covered in [Statistical inference](#) section.

³The analytic solution for the central tendency estimate that minimizes `SD` is, of course, the sample `mean` ($\frac{1}{n} \sum_{i=1}^n y_i$).

some there is no analytic solution and *computational* approaches must be utilized. These computation approaches are referred to as *optimization algorithms*. The example given here involves only one parameter that needs to be optimized, in this case body height estimate, but real-life problems involve numerous parameters. The simple search through parameters *state-space* would take forever when it comes to problems involving more than only a few parameters. Algorithms that solve this computational problems are numerous, out of which the most popular ones are *gradient descent*, and *Markov Chain Monte-Carlo* (MCMC), which is utilized in *Bayesian inference* (will be discussed in *Bayesian perspective* section).

The take-home message from this short interlude is that even the simple descriptive statistics can be seen as statistical models.

If we take another cost function, for example *mean absolute error* (**MAE**) (Equation (2.6)) and if we *optimize* so that the sample central tendency estimate minimizes MAE, we will get **median** estimator.

$$MAE = \frac{1}{n} \sum_{i=1}^n | \hat{y}_i - y_i | \quad (2.6)$$

We will expand this discussion about loss functions, cost functions, and performance metrics in [Sample mean as the simplest predictive model](#) section. For more information please check the package *Metrics* (Hamner and Frasco 2018) and the following references (Botchkarev 2019; Chai and Draxler 2014; Willmott and Matsuura 2005; Barron 2019).

2.1.2 Effect Sizes

Besides describing groups, we are often interested in comparing them. In order to achieve this task, a collection of estimators termed *effect size statistics* are utilized. Effect size can be defined in a *narrow sense* or in a *broad sense*. Briefly, the narrow sense refers to a family of standardized measures such as Cohen's **d**, while the broad sense refers to any measure of interest, standardized or not. The approach to effect size statistics in this book is thus in a broad sense of the definition, in which all group comparison estimators are considered effect sizes statistics. In order to estimate effect sizes, one group needs to be considered *baseline* or *control*. The most common effect size statistics can be found in the Table 2.2 where female body height is considered baseline and compared with male body height.

Difference, or **mean difference** (**mean diff**) is calculated by subtracting group **means**. Using body height as an example, the **mean diff** between males and females is calculated by using the following equation (2.7):

Table 2.2: Effect size statistics for estimating differences between two independent groups

Difference (cm)	SDdiff (cm)	% CVdiff	% Difference	Ratio	Cohen's d	CLES	OVL
12.73	12.41	97.55	7.8	1.08	1.45	0.85	0.47

$$\begin{aligned}
 mean_{difference} &= mean_{males} - mean_{females} \\
 mean_{males} &= \frac{1}{n} \sum_{i=1}^n male_i \\
 mean_{females} &= \frac{1}{n} \sum_{i=1}^n female_i
 \end{aligned} \tag{2.7}$$

% CVdiff, or percent coefficient of variation of the difference is the standard deviation of the difference (SDdiff - explained shortly) divided by mean diff (Equation (2.8)):

$$\% CV_{difference} = 100 \times \frac{SD_{difference}}{mean_{difference}} \tag{2.8}$$

% Difference, or mean percent difference is calculated by dividing mean diff with the mean of the baseline group, in this case the female group, multiplied by 100 (Equation (2.9)):

$$mean\%_{difference} = 100 \times \frac{mean_{difference}}{mean_{females}} \tag{2.9}$$

Mean ratio, as its name suggests, is simple ratio between the two means (Equation (2.10)):

$$mean_{ratio} = \frac{mean_{males}}{mean_{females}} \tag{2.10}$$

Cohen's d represent standardized effects size and thus preferable effect size statistic. For this reason, Cohen's d is commonly written as ES, short of effect size. Cohen's d for the independent groups is calculated by dividing mean diff (Equation (2.7)) with pooled standard deviation ((2.11)).

$$Cohen's\ d = \frac{mean_{difference}}{SD_{pooled}} \tag{2.11}$$

Pooled standard deviation represents combined standard deviations from two groups (Equation (2.12)).

Table 2.3: **Training intervention effect sizes for YoYoIR1 and 30-15IFT.**
Modified from Buchheit and Rabbani (2014)

Test	Pre-training	% Change	Cohen's d
YoYoIR1	1031 \pm 257 m	35 %	1.2
30-15IFT	17.4 \pm 1.1 kmh/h	7 %	1.1

$$SD_{pooled} = \sqrt{\frac{(n_{males} - 1)SD_{males}^2 + (n_{females} - 1)SD_{females}^2}{n_{males} + n_{females} - 2}} \quad (2.12)$$

Why Cohen's d should be used instead of other effect size estimators can be demonstrated by a simple example, coming from a study by Buchheit and Rabbani (2014). In this study, authors examined the relationship between the performance in the *YoYo Intermittent Recovery Test Level 1* (YoYoIR1) and the *30-15 Intermittent Fitness Test* (30-15IFT), and compared the *sensitivity* of both tests to the training. Although this study used two dependent groups (Pre-training and Post-training), the rationale can be applied to the topic of estimating effect sizes between the two independent groups. Table 2.3 contains Pre-training results and the effect sizes estimated with `percent change`⁴ and Cohen's d.

Since YoYoIR1 and 30-15IFT utilize different scales (total meters covered and velocity reached respectively), `percent change` estimator is not a good choice to compare the effect sizes between the two tests⁵. Since Cohen's d is standardized estimator, it should be used when comparing tests or measures that are at different scales.

After estimating effect sizes, the question that naturally follows up is the question of *magnitude*. In other words - "how big is the effect?". Since Cohen's d is standardized estimator, it allows for establishment of qualitative magnitude thresholds. Based on the original work by Cohen (Cohen 1988), Hopkins (Hopkins 2006; Hopkins et al. 2009) suggested the following magnitudes of effect (Table 2.4). According to the Table 2.4, the body height difference between males and females would be considered *large*, as well as changes in both YoYoIR1 and 30-15IFT.

Cohen's d, as well as associated magnitudes of effect, are commonly hard to interpret by non-statistically trained professionals (e.g. coaches). McGraw and Wong (1992) suggested *common language effect size* (CLES) estimator instead,

⁴Percent change is the same estimator as percent difference, but applied to difference between the two dependent groups (see section Comparing dependent groups).

⁵However, let's admit that we would rather report estimators of higher value, particularly if we are biased toward a specific test. "Athletes improved on average for 35%" sounds much more appealing than 7%, even if the effects estimated using Cohen's d are the same.

Table 2.4: Magnitudes of effect

Magnitude of effect	Trivial	Small	Moderate	Large	Very Large	Nearly Perfect
Cohen's d	0 - 0.2	0.2 - 0.6	0.6 - 1.2	1.2 - 2.0	2.0 - 4.0	> 4.0

which could be more intuitive to understand. CLES represents the probability that an observation sampled at random from one group will be greater than an observation sampled at random from other group. For example, if we take random male and random female from our two groups and repeat that 100 times⁶, how many times a male would be taller than a female (Figure 2.3)?

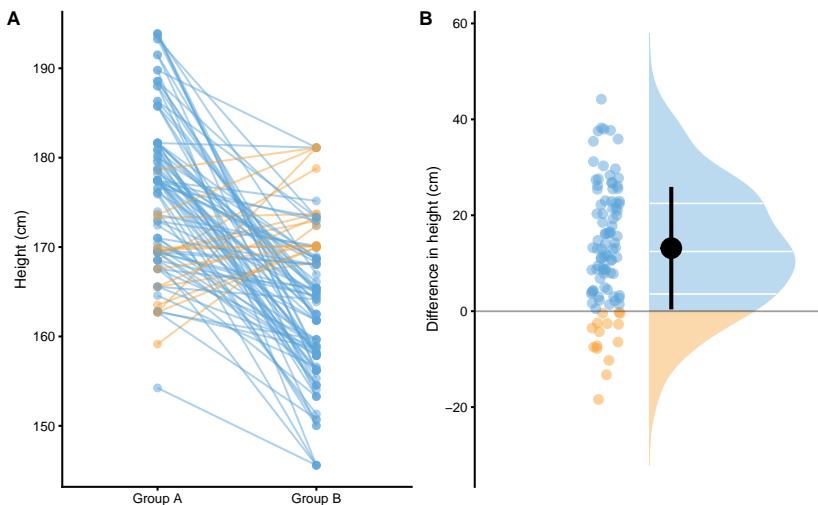


Figure 2.3: Drawing random 100 pairs to estimate probability of males being taller than females. **A.** Scatterplot of 100 pairs drawn at random from two samples. Since we are comparing paired males and females, lines can be drawn between each of 100 draws. Blue line indicates taller male, while orange line indicates taller female. **B.** Distribution of the difference between males and females for each of 100 pairs drawn

By using simple counting from 100 random paired samples, males are taller in 85 cases, or 85%. By using probability, that is equal to 0.85. In other words, if I blindfoldedly, randomly select a male and a female from the two groups and if I bet that the male is taller, I would be correct 85% of the time.

CLES can be estimated using *brute-force* computational method, or *algebraic* method. Brute-force method involves generating all possible pair-wise combi-

⁶In other words, we are drawing 100 paired samples from the two independent groups. This makes the drawn 100 observations paired or dependent.

nations from two groups, and in our example that is equal to $50 \times 50 = 2500$ cases, and then simply counting in how many cases males are taller than females. This method can become very computationally intensive for groups with large sample number. Algebraic method, on the other hand, assumes normal distribution of the observations in the groups, and estimates *standard deviation of the difference (SDdiff)* (Equation (2.13)). Note that standard deviation of the all pairwise differences estimated with brute-force method would be very similar to algebraically derived SDdiff.

$$SD_{difference} = \sqrt{SD_{males}^2 + SD_{females}^2} \quad (2.13)$$

Algebraically, CLES is then derived assuming normal distribution (where mean of the distribution is equal to `mean diff` between the groups, and standard deviation of the distribution is equal to `SDdiff`) by calculating probability of the difference scores higher than zero (see Figure 2.3B for a visual representation). Table 2.2 contains algebraically computed CLES estimate.

CLES equivalent is utilized as a performance metric in class prediction tasks, termed *area under curve (AUC)*, where 0.5 is a predictive performance equal to a random guess, and 1 is perfect predictive separation between the two classes (James et al. 2017; Kuhn and Johnson 2018).

Overlap (OVL) estimator represents the overlap between the two sample distributions. Providing that samples are identical, the OVL is equal to 1. Providing there is complete separation between the two samples, then OVL is equal to 0 (Figure 2.4A). OVL can be estimated with brute-force computational methods (which doesn't make assumptions regarding sample distribution) and with algebraic methods that make normality assumptions.

Since Cohen's *d*, CLES and OVL are mathematically related, it is possible to convert one to another (assuming normal distribution of the samples and equal SD between the two groups for the OVL estimation). Figure 2.4B depicts relationship between the Cohen's *d*, CLES, and OVL. Figure 2.4C depicts relationship between the CLES and OVL.

Table 2.5: Magnitudes of effect for CLES and OVL estimated using Cohen's d

Magnitude of effect	Trivial	Small	Moderate	Large	Very Large	Nearly Perfect
Cohen's d	0 - 0.2	0.2 - 0.6	0.6 - 1.2	1.2 - 2.0	2.0 - 4.0	> 4.0
CLES	0.50 - 0.56	0.56 - 0.66	0.66 - 0.80	0.80 - 0.92	0.92 - 1.00	1.00
OVL	1.00 - 0.92	0.92 - 0.76	0.76 - 0.55	0.55 - 0.32	0.32 - 0.05	0.00

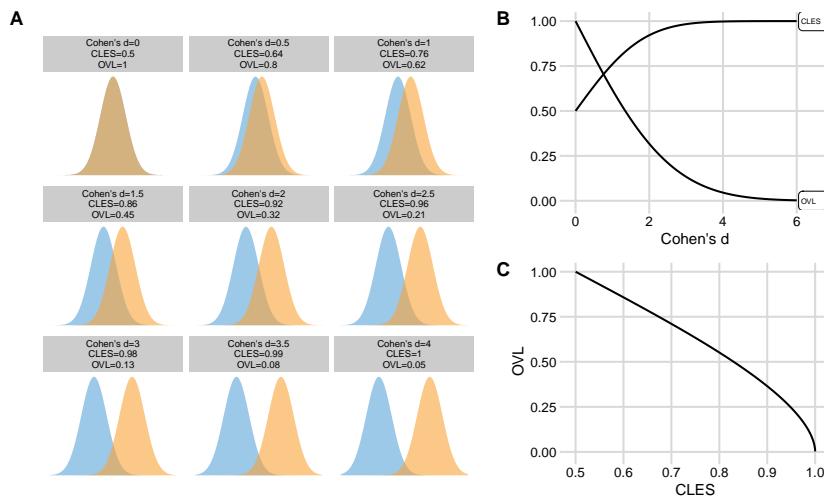


Figure 2.4: Relationship between the Cohen's d, CLES, and OVL. **A.** Visual display of the samples of varying degrees of separations, and calculated Cohen's d, CLES, and OVL. **B.** Relationship between the CLES and OVL to the Cohen's d. **C.** Relationship between the CLES and OVL

Table 2.5 contains Cohen's d magnitudes of effect with accompanying estimated CLES and OVL thresholds.

2.1.3 The Smallest Effect Size Of Interest

According to Cohen (1988), the qualitative magnitude thresholds from Table 2.5 are “arbitrary conventions, recommended for use only when no better basis for estimating the effect size is available” (p. 12). But what if practitioners *a priori* know what is the *minimal important* effect size and are interested in judging the *practical or clinical significance* (Sainani 2012) of the results (in this case difference between the groups)? In other words, the *smallest effect size of interest* (SESOI)⁷.

⁷Other term for SESOI that is commonly used is *region of practical equivalence* (ROPE) (Kruschke and Liddell 2018a, 2018b).

There is no single way to approach definition and estimation of SESOI, but it usually tends to be based on either the known *measurement error* (ME) (e.g. the minimum *detectable* effect size), or the effect size that is large enough to be practically meaningful (e.g. the minimal *important* difference, or the smallest worthwhile change) (Anvari and Lakens 2019; Will G Hopkins 2004a; Will G. Hopkins 2015; King 2011; Lakens, Scheel, and Isager 2018; Turner et al. 2015; Swinton et al. 2018; Caldwell and Cheuvront 2019). In this book, statistical models and estimators that utilize SESOI are referred to as *magnitude-based*.

To introduce magnitude-based estimators, consider $\pm 2.5\text{cm}$ to be body height SESOI⁸, or the difference that would be practically significant. In other words, individuals with height difference within $\pm 2.5\text{cm}$ would be considered practically equivalent (from the minimal important effect perspective), or it might be hard to detect this difference with a quick glance (from minimum detectable effect perspective).

The simplest magnitude-based statistics would be `mean diff` divided by SESOI (`Difference to SESOI`) (Equation (2.14)). This estimator, similar to other standardized estimators (e.g. Cohen's *d*) allows comparison of variables at different scales, but it would also give more insight into differences from practical significance perspective.

$$\text{Difference to SESOI} = \frac{\text{mean difference}}{\text{SESOI}_{\text{upper}} - \text{SESOI}_{\text{lower}}} \quad (2.14)$$

Second magnitude-based statistic is `SDdiff` divided by SESOI (`SDdiff to SESOI`) (Equation (2.15)). This estimator, similar to `% CVdiff`, would answer how variable are the differences compared to SESOI.

$$\text{SDdiff to SESOI} = \frac{\text{SD difference}}{\text{SESOI}_{\text{upper}} - \text{SESOI}_{\text{lower}}} \quad (2.15)$$

Similarly, `CLES` estimator can become magnitude-based by utilizing SESOI. Rather than being interested in probability of a random male being taller than a random female (out of the two sample groups), we might be interested in estimating how probable are *lower*, *equivalent*, and *higher* (or usually defined as *harmful*, *trivial*, and *beneficial*) differences defined by SESOI. Practically equivalent (trivial) differences are differences ranging from $\text{SESOI}_{\text{lower}}$ to $\text{SESOI}_{\text{upper}}$, while everything over $\text{SESOI}_{\text{upper}}$ is higher (or beneficial) difference and everything lower than $\text{SESOI}_{\text{lower}}$ is lower (or harmful) difference.

Using brute-force computational method and drawing all pair-wise combinations from the two groups ($50 \times 50 = 2500$ cases), and using $\pm 2.5\text{cm}$ SESOI as a *prac-*

⁸SESOI has two thresholds: *lower* and *upper*, or negative and positive. In this example these thresholds are -2.5cm and $+2.5\text{cm}$. This makes SESOI range equal to 5cm , which is calculated as $\text{SESOI}_{\text{upper}} - \text{SESOI}_{\text{lower}}$. This range can also be referred to as *equivalence range*.

tically equivalent difference⁹, we can estimate probabilities of lower (*pLower*), equivalent (*pEquivalent*) and higher difference (*pHigher*) by calculating proportion of cases within each magnitude band (Figure 2.5).

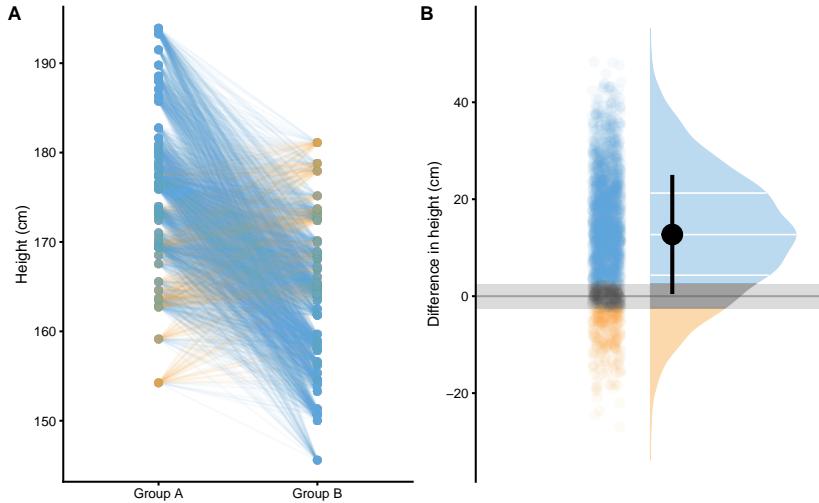


Figure 2.5: Pairwise comparison of males and females to estimate probability of lower, equivalent, and higher magnitude of difference. **A.** Scatterplot of all pair-wise combinations ($50 \times 50 = 2500$), drawn at random out of two samples. Since we are comparing paired males and females, lines can be drawn between each of 2500 draws. Blue line indicates males taller than females higher than SESOI, equivalent lines indicates pairs with a height difference less or equal to SESOI, while orange line indicates females taller than males higher than SESOI. **B.** Distribution of the differences between males and females for all 2500 pair-wise combinations. Grey band indicates SESOI. Surface of the distribution over SESOI (blue color) indicates probability of randomly selected male being taller than a randomly selected female (*pHigher*), with a height difference of at least SESOI magnitude. Surface of the distribution under SESOI (orange color) indicates probability of randomly selected female being taller than a randomly selected female (*pLower*), with a height difference of at least SESOI magnitude. Grey surface area indicates probability of randomly selecting male and female with a height difference within SESOI band (*pEquivalent*)

Table 2.6 contains estimated probabilities of observing lower, equivalent, and higher differences in height between the randomly selected male and female using brute-force computational method and algebraic method. These estimates answer the following question “If I compare random male and random female from

⁹It is assumed here that SESOI is *symmetrical* in both positive and negative directions. This makes the equivalent difference ranging from -2.5cm to +2.5cm. SESOI doesn't necessarily need to be symmetrical in both positive and negative directions.

Table 2.6: Estimated probabilities of observing lower, equivalent, and higher differences in height

Method	pLower	pEquivalent	pHigher
brute-force	0.110	0.096	0.794
algebraic	0.111	0.095	0.794

Table 2.7: Magnitude-based effect size statistics for estimating difference between two independent groups

SESOI lower (cm)	SESOI upper (cm)	Difference to SESOI	SDdiff to SESOI	pLower	pEquivalent	pHigher
-2.5	2.5	2.55	2.48	0.11	0.09	0.79

my sample, how probable are lower/equivalent/higher magnitudes of difference in height?”. Asking such a magnitude-based question regarding the random individual difference represents a form of prediction question and predictive task. In this book, such questions are answered with *magnitude-based prediction* approaches.

It is common to represent means as *systematic component* or *fixed effect* (e.g. `mean difference`), and variability around the mean (i.e. `SDdiff`) as *stochastic component* or *random effect*. It is unfortunate that the common statistical modeling and analysis, particularly in sport science, takes the stance of approaching and treating between-individual variation as *random error*. The approach suggested in this book complements *group-based* or *average-based* statistics with magnitude-based predictions that aim to help in answering individual-based questions, common to sport practitioners. Table 2.7 contains discussed magnitude-based estimators that can complement common effect size statistics (Table 2.2) when comparing two independent groups.

2.2 Comparing dependent groups

As an example of dependent or paired groups descriptive analysis, let’s consider the simple *Pre-test* and *Post-test* design. We have given training intervention to a group of N=20 males involving bench-press training. Training intervention involved performing bench pressing two times a week for 16 weeks. One-repetition-maximum (1RM) in the bench press was performed before (Pre-test) and after (Post-test) training intervention. Table 2.8 contains individual Pre-test and Post-test scores, as well as the Change in the bench press 1RM.

The results of this simple Pre-test and Post-test design can be described in multiple ways. Here, I will present the three most common approaches.

Table 2.8: Individual Pre and Post scores, as well as Change in the bench press 1RM

Athlete	Pre-test (kg)	Post-test (kg)	Change (kg)
Athlete 01	111.80	121.42	9.62
Athlete 02	95.95	102.13	6.18
Athlete 03	105.87	125.56	19.69
Athlete 04	98.79	109.67	10.87
Athlete 05	95.81	108.11	12.30
Athlete 06	95.27	92.67	-2.60
Athlete 07	97.75	106.03	8.28
Athlete 08	106.50	109.51	3.01
Athlete 09	80.62	95.96	15.34
Athlete 10	100.40	94.30	-6.11
Athlete 11	82.71	78.91	-3.80
Athlete 12	102.89	93.98	-8.91
Athlete 13	91.34	105.21	13.87
Athlete 14	111.14	108.07	-3.07
Athlete 15	95.13	96.01	0.88
Athlete 16	109.12	112.12	3.00
Athlete 17	91.87	103.41	11.54
Athlete 18	92.16	103.93	11.77
Athlete 19	108.88	119.72	10.84
Athlete 20	97.94	95.91	-2.03

Table 2.9: Descriptive analysis of the Pre-test, Post-test, and Change as independent samples

Estimator	Pre-test	Post-test	Change
n	20.00	20.00	20.00
mean (kg)	98.60	104.13	5.53
SD (kg)	8.70	11.08	8.05
% CV	8.83	10.64	145.46
median (kg)	97.84	104.57	7.23
MAD (kg)	8.64	11.94	8.46
IQR (kg)	11.64	13.60	13.77
mode (kg)	96.49	105.76	10.78
min (kg)	80.62	78.91	-8.91
max (kg)	111.80	125.56	19.69
range (kg)	31.18	46.64	28.60
skew	-0.26	-0.05	-0.16
kurtosis	-0.73	-0.28	-1.28

2.2.1 Describing groups as independent

The simplest analysis involve descriptive statistics assuming groups as independent. Table 2.9 contains descriptive statistics applied to Pre-test, Post-test and Change scores as independent. Figure 2.6 visualizes the scores using three raincloud plots.

Table 2.10: **Effect size statistics for estimating change in two dependent groups**

Change (kg)	SDchange (kg)	% CVchange	% Change	Ratio	Cohen's d	CLES	OVL
5.53	8.05	145.46	5.75	1.06	0.64	0.65	0.75

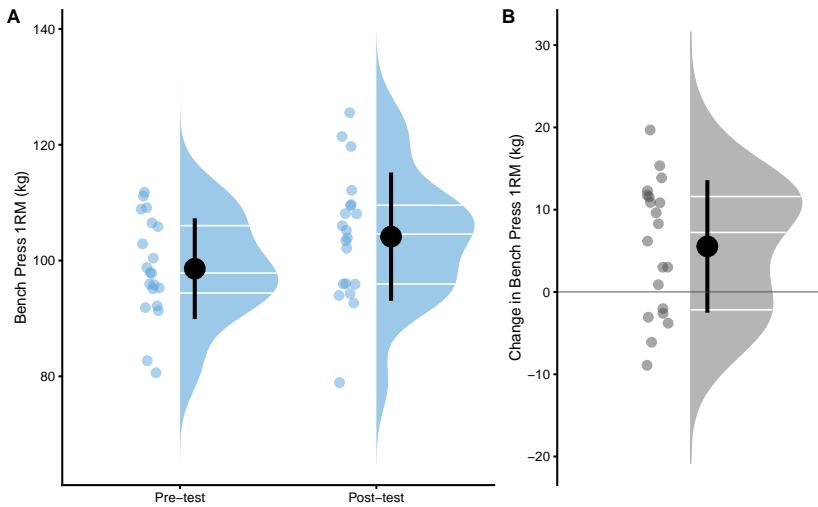


Figure 2.6: **Raincloud plots of the Pre-test, Post-test and Change scores in the bench press 1RM. A.** Distribution of the Pre-test and Post-test scores. **B.** Distribution of the Change score

2.2.2 Effect Sizes

Table 2.10 contains the most common effect size estimators utilized when describing change in the Pre-Post paired design. The terminology utilized in this book differentiates between the *difference* which is used in independent groups and the *change* which is used in paired or dependent groups

Change, or mean change is calculated by taking average of the change score (Equation (2.16)). Change score is simple difference between Pre-test and Post-test.

$$\begin{aligned}
 mean_{change} &= \frac{1}{n} \sum_{i=1}^n (post_i - pre_i) \\
 mean_{change} &= \frac{1}{n} \sum_{i=1}^n change_i \\
 change_i &= post_i - pre_i
 \end{aligned} \tag{2.16}$$

SDchange, or standard deviation of the change is a simple standard deviation of the change (Equation (2.17)). It represents a measure of dispersion of the change scores.

$$SD_{change} = \sqrt{\frac{1}{n-1} \sum_{i=1}^n (change_i - mean_{change})^2} \quad (2.17)$$

% **CVchange**, or percent coefficient of variation of the change is the **SDchange** divided by **mean change** (Equation (2.18)).

$$\% CV_{change} = 100 \times \frac{SD_{change}}{mean_{change}} \quad (2.18)$$

% **Change**, or **Mean percent change** is calculated by taking a mean of the ratio between the change and the Pre-test, multiplied by 100 (Equation (2.19)).

$$mean\%_{change} = 100 \times \frac{1}{n} \sum_i^n \frac{change_i}{pre_i} \quad (2.19)$$

Mean ratio represents mean of the Post-test to Pre-test scores ratios (Equation (2.20)).

$$mean_{ratio} = \frac{1}{n} \sum_i^n \frac{post_i}{pre_i} \quad (2.20)$$

Cohen's d represents standardized effect size of the change. In the paired design, **Cohen's d** is calculated by dividing **mean change** with standard deviation of the Pre-test scores (**SDpre**) (Equation (2.21)).

$$Cohen's\ d = \frac{mean_{change}}{SD_{pre}} \quad (2.21)$$

CLES for the paired groups represents probability of observing positive change. **OVL**, equally to the independent groups, represents overlap between the Pre-test and Post-test scores.

Magnitude-based effect size estimators involve the use of SESOI and can be found on Table 2.11. Similarly to magnitude-based effect size estimators with the independent groups, magnitude-based effect size estimators with the paired group involve **Change to SESOI**, **SDchange to SESOI** as well as proportions of lower (**pLower**), equivalent (**pEquivalent**) and higher (**pHigher**) change scores.

Figure 2.7 depicts visually how proportions of lower, equivalent, and higher change scores are estimated. Same as with two independent groups, these proportions can be estimated using the brute-force method (i.e. simple counting of the change scores within lower, trivial, and higher zones), or algebraic where

Table 2.11: Magnitude-based effect size statistics for estimating change between two dependent groups

SESOI (kg)	Change to SESOI	SDchange to SESOI	pLower	pEquivalent	pHigher
±5	0.55	0.81	0.1	0.37	0.53

SDchange is utilized and assumption of the normally distributed change scores is made.

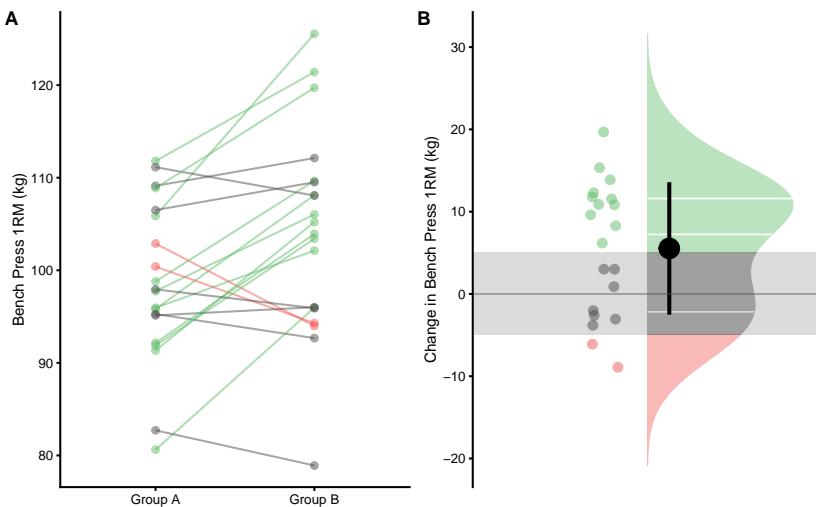


Figure 2.7: Visual analysis of the dependent groups scores using SESOI. **A.** Scatter plot of Pre-test and Post-test scores. Green line indicates change higher than SESOI upper, grey line indicates change within SESOI band, and red line indicates negative change lower than SESOI lower. **B.** Distribution of the change scores. Green area represents proportion of change scores higher than SESOI upper, red area represents proportion of negative change scores lower than SESOI lower, and grey area indicates equivalent change, which is within SESOI band

It might be tempting to claim that this intervention is *causing* changes in the bench press 1RM, but we should be wary of doing that. It is important to keep in mind that the effect size estimators are used only descriptively without any causal connotation. To make causal claims, further criteria need to be taken into account. This is discussed in more details in the [Causal inference](#) section of this book.

2.3 Describing relationship between two variables

So far, we have dealt with single variable descriptive statistics. However, we are often interested in relationship or *association* between two variables. One of these variables takes the role of the *dependent variable* (*outcome* or *target variable*) and the other of the *independent variable* (or *predictor variable*).

Let's assume we tested N=30 female soccer athletes by using two tests: (1) YoYoIR1 test (expressed in meters), and (2) *maximum aerobic speed* (MAS) test (expressed in km/h)¹⁰. Variables in this example represent observations in each test (Table 2.12).

Descriptive statistics for YoYoIR1 and MAS test results can be found in the Table 2.13.

Visual analysis in Figure 2.8 depicts the association between these two tests using scatter plot.

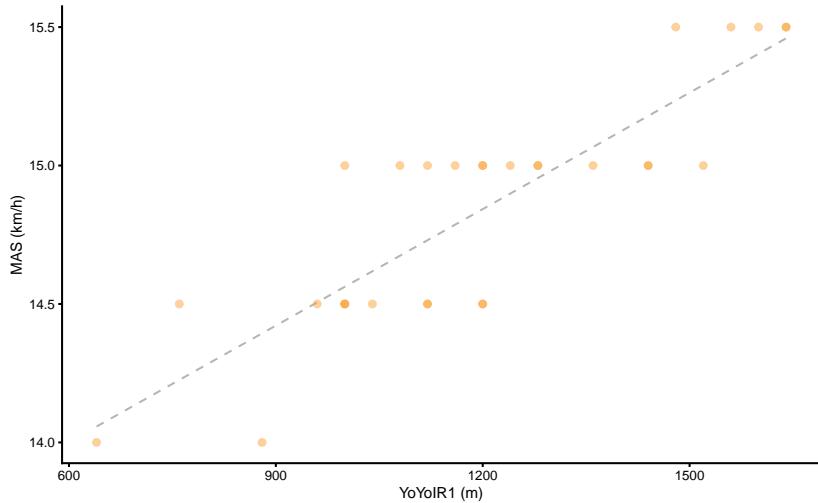


Figure 2.8: **Scatter plot between two variables.** Dashed line represents linear regression line

Table 2.14 contains common estimators of the association between two variables. All estimators except *maximum information coefficient* (MIC) (Albanese et al. 2012b; Reshef et al. 2011) assumes linear relationship between two variables. It is thus important to visually analyze the association (see Figure 2.8) before trusting numerical estimators.

¹⁰Since YoYoIR1 test is performed in 2x20m shuttles, the minimal increment is equal to 40m. For the MAS test the minimal increment is 0.5km/h.

Table 2.12: **Results of YoYoIR1 and MAS tests for N=30 female soccer athletes**

Athlete	YoYoIR1 (m)	MAS (km/h)
Athlete 01	1640	15.5
Athlete 02	1080	15.0
Athlete 03	1440	15.0
Athlete 04	1200	15.0
Athlete 05	960	14.5
Athlete 06	1120	15.0
Athlete 07	1000	14.5
Athlete 08	1440	15.0
Athlete 09	640	14.0
Athlete 10	1360	15.0
Athlete 11	760	14.5
Athlete 12	1240	15.0
Athlete 13	1000	15.0
Athlete 14	1600	15.5
Athlete 15	1160	15.0
Athlete 16	1520	15.0
Athlete 17	1000	14.5
Athlete 18	1000	14.5
Athlete 19	1480	15.5
Athlete 20	1280	15.0
Athlete 21	1200	14.5
Athlete 22	1200	14.5
Athlete 23	1200	15.0
Athlete 24	1120	14.5
Athlete 25	1560	15.5
Athlete 26	1120	14.5
Athlete 27	1640	15.5
Athlete 28	1280	15.0
Athlete 29	1040	14.5
Athlete 30	880	14.0

Table 2.13: Descriptive statistics for YoYoIR1 and MAS test results

Estimator	YoYoIR1	MAS
n	30.00	30.00
mean	1205.33	14.85
SD	255.96	0.42
% CV	21.24	2.82
median	1200.00	15.00
MAD	296.52	0.74
IQR	410.00	0.50
mode	1131.68	15.00
min	640.00	14.00
max	1640.00	15.50
range	1000.00	1.50
skew	-0.02	-0.11
kurtosis	-0.68	-0.72

Table 2.14: Common estimators of the association between two variables

Pearson r	R-squared	MIC
0.86	0.74	0.55

The *Pearson product-moment correlation coefficient* (**Pearson's r**) is a measure of the strength of the linear relationship between two variables (Equation (2.22)).

$$r = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^n (x_i - \bar{x})^2 (y_i - \bar{y})^2}} \quad (2.22)$$

Pearson's r is standardized measure that can take values ranging from -1 to +1, where 0 indicates no relationship, and -1 and +1 indicates perfect relationship. Negative **Pearson's r** value represents negative association (i.e. as one variable increases the other decreases), while positive **Pearson's r** value represents positive association (i.e., as one variable increases so does the other).

R-squared (R^2) represents *variance explained*, i.e. how much the *model* explains variance in the target variable. In this example the model is *linear regression*. **R-squared** is standardized measure of association that can take values ranging from zero (no association, or no variance explained) to 1 (perfect association, or all variance explained). **R-squared**, as its name suggests, represents Pearson's r squared, but for more complex models it can be calculated using variances or *mean squares* (MS) (Equation (2.22)):

$$\begin{aligned} R^2 &= \frac{MS_{model}}{MS_{total}} \\ MS_{model} &= \frac{1}{n} \sum_{i=1}^n (\hat{y}_i - \bar{y})^2 \\ MS_{total} &= \frac{1}{n} \sum_{i=1}^n (y_i - \bar{y})^2 \end{aligned} \quad (2.23)$$

Maximal information coefficient (**MIC**) is a novel measure of the strength of the linear or non-linear association between two variables and belongs to the *maximal information-based non-parametric exploration* (MINE) class of statistics (Albanese et al. 2012b; Reshef et al. 2011). **MIC** is standardized measure of association that can take values ranging from zero (no association) to 1 (perfect association). As opposed to **Pearson r**, **MIC** can *pick up* non-linear association between two variables.

Statistical model, or *machinery* underlying **Pearson r** and **R-squared** is linear regression. Similar to a sample **mean** (see section [Sample mean as the simplest statistical model](#)), linear regression can be seen as *optimization algorithm* that tries to find a line that passes through the data with the minimal error.¹¹ A solution to this problem can be found computationally or analytically¹². Either way, the *coefficients* (or *parameters*) that need to be estimated in this example

¹¹This approach, as already explained, belongs to the OLS approach. On the other hand, MLE tries to find a line that maximizes likelihood of the data.

¹²The benefit of using *squared errors* in OLS approaches, is that this *optimization* (or the search for parameters that minimize **RMSE** as a cost function in this case) can be done analytically. One of the drawbacks of using squared errors (or squared residuals) is sensitivity

Table 2.15: Linear regression estimates for `intercept`, `slope coefficient`, and `RSE` when MAS is the target variable and YoYoIR1 is the predictor

Intercept (km/h)	Slope	RSE (km/h)
13.16	0.0014	0.22

with two variables are `intercept` ($\hat{\beta}_0$), `slope coefficient` ($\hat{\beta}_1$), and `residual error` ($\hat{\epsilon}$) (Equation (2.24)).

$$\hat{y}_i = \hat{\beta}_0 + \hat{\beta}_1 x_i + \hat{\epsilon} \quad (2.24)$$

Table 2.15 contains estimates for `intercept`, `slope`, and residual error. Residual error (ϵ) is estimated by using `residual standard error` (`RSE`), which is similar to already discussed `RMSE`, but rather than dividing sum of square errors by n observations, it is divided by $n - p$ (Equation (2.25)). The p is the number of model parameters, in this case 2 (`intercept` and one `slope coefficient`).

$$RSE = \sqrt{\frac{1}{n-p} \sum_{i=1}^n (y_i - \hat{y}_i)^2} \quad (2.25)$$

Estimated parameters in the Table 2.15 can be written using the linear equation format (Equation (2.26)).

$$MAS = 13.16 + 0.0014 \times YoYoIR1 \pm 0.22 \text{ km/h} \quad (2.26)$$

Slope coefficient of 0.0014 can be interpreted the following way: if YoYoIR1 increases by 500m, then MAS would increase by 500×0.0014 or 0.7km/h.

Although measures of association between two variables, such as `Pearson's r` and `R-squared`, are symmetrical (meaning it doesn't matter which variable is predictor or target), one cannot reverse the linear regression equation to get YoYoIR1 from MAS as done in the Equation (2.25).

$$\begin{aligned} MAS &= \hat{\beta}_0 + \hat{\beta}_1 \times YoYoIR1 \\ YoYoIR1 &= \frac{-\hat{\beta}_0 + MAS}{\hat{\beta}_1} \\ YoYoIR1 &= -\frac{\hat{\beta}_0}{\hat{\beta}_1} + \frac{1}{\hat{\beta}_1} \times MAS \\ YoYoIR1 &= -9385.59 + 713.19 \times MAS \end{aligned} \quad (2.27)$$

to *outliers*. Other regression approaches, such as *quantiles regression* or *ordinary least products* (OLP) for example, use different loss and cost functions. OLP regression will be utilized in the `Reliability` section of the book.

Table 2.16: Linear regression estimates for `intercept`, `slope coefficient`, and `RSE` when YoYoIR1 is the target variable and MAS is the predictor

Intercept (m)	Slope	RSE (m)
-6589.82	524.93	133.84

It can be seen that the reverse parameters from (2.25) differ from the parameters in the Table 2.16 which are estimated using YoYoIR1 as the target variable and MAS as the predictor variable.

This difference between reversed parameters and correctly estimated can be visually seen as non-identical linear regression lines in the Figure 2.9.

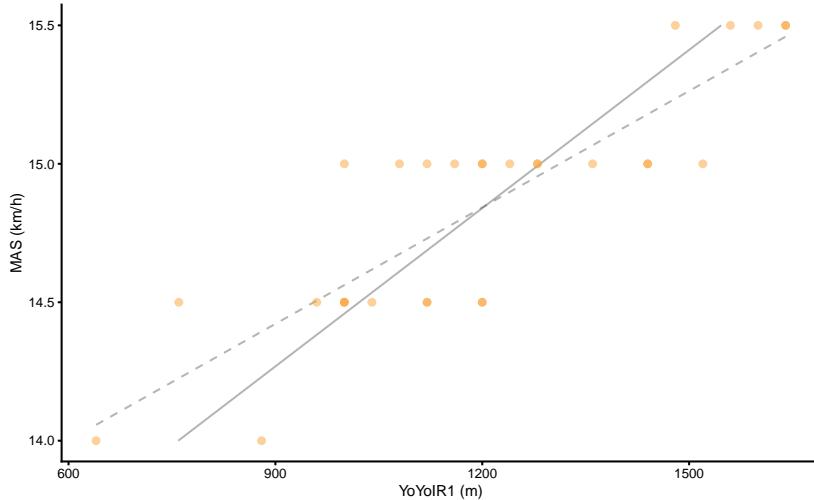


Figure 2.9: **Regression line differs depending which variable is target or the outcome variable.** Dashed grey line represents regression line when MAS is the target variable. Grey line represents regression line when YoYoIR1 is the target variable. Since they are not identical, one cannot reverse the equation to predict YoYoIR1 from MAS score, when such equation is estimated by predicting MAS from YoYoIR1

Unfortunately, this is common practice in sport science. Rather than reversing parameters, one needs to fit, in this case, linear regression model again with the properly defined target and predictor variables. In certain scenarios, such as [Reliability](#) analysis, we do not know which variable represents predictor and which represents target or outcome. For this reason, different approaches to regression, such as *ordinary least products* (OLP) are utilized (Ludbrook 2010, 2012, 1997, 2002; Mullineaux, Barnes, and Batterham 1999). These topics will

be covered in the second part of this book.

2.3.1 Magnitude-based estimators

Similarly to independent and dependent group analysis, with association we might be interested in the practical significance of the results. In order to judge results from practical significance perspective, we need to define SESOI of both variables (i.e. YoYoIR1 and MAS). Using minimal test increment, SESOI for the YoYoIR1 test is defined as ± 40 m, and SESOI for the MAS test is defined as ± 0.5 km/h.

One question we might ask is whether the YoYoIR1 SESOI is associated with MAS SESOI. This can be answered with the **sensitivity** estimator (Equation (2.28)).

$$\begin{aligned}
 \text{Sensitivity} &= \frac{(SESOI_{YoYoIR1_{upper}} - SESOI_{YoYoIR1_{lower}}) \times \hat{\beta}_1}{SESOI_{MAS_{upper}} - SESOI_{MAS_{lower}}} \\
 \text{Sensitivity} &= \frac{(40 - -40) \times 0.0014}{0.5 - -0.5} \\
 \text{Sensitivity} &= \frac{(80) \times 0.0014}{1} \\
 \text{Sensitivity} &= \frac{0.11}{1} \\
 \text{Sensitivity} &= 0.11
 \end{aligned} \tag{2.28}$$

This means that the change in the YoYoIR1 test equal to SESOI will yield only a small proportion of SESOI in the MAS test.

In the case where SESOI of the MAS test is unknown, using known SESOI of the YoYoIR1 test can be used to estimate it. This is done by using estimated $\hat{\beta}_1$ (**slope coefficient**), as demonstrated in the Equation (2.29).

$$\begin{aligned}
 SESOI_{MAS_{upper}} &= \hat{\beta}_1 \times SESOI_{YoYoIR1_{upper}} \\
 SESOI_{MAS_{upper}} &= 0.0014 \times 40 \\
 SESOI_{MAS_{upper}} &= 0.06 \text{ km/h} \\
 \\
 SESOI_{MAS_{lower}} &= \hat{\beta}_1 \times SESOI_{YoYoIR1_{lower}} \\
 SESOI_{MAS_{lower}} &= 0.0014 \times -40 \\
 SESOI_{MAS_{lower}} &= -0.06 \text{ km/h}
 \end{aligned} \tag{2.29}$$

Next magnitude-based question might be related to the practically significant strength of the association between two variables. For example, we would like to

know if the residuals are higher or lower than the SESOI in the target variable (i.e. MAS, which is equal to $\pm 0.5\text{km/h}$). Figure 2.10 depicts scatter plot between two variable (panel A) and residuals (panel B) utilizing SESOI in MAS as the grey area.

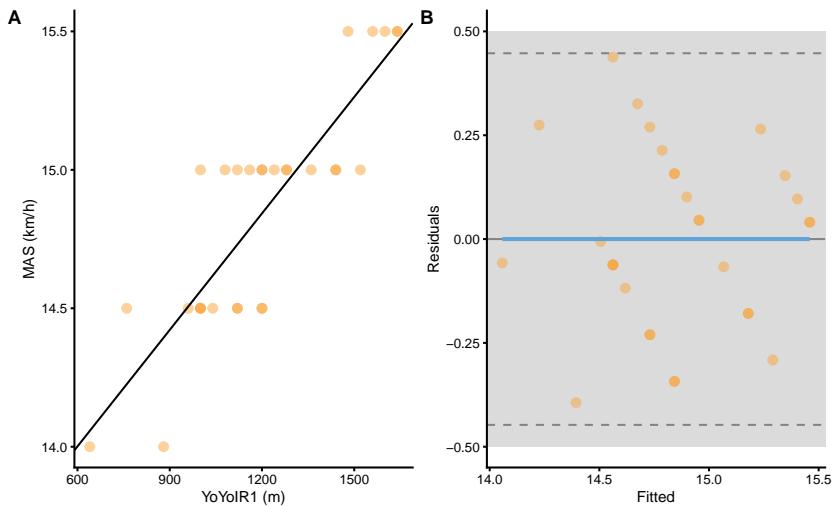


Figure 2.10: **Scatter plot between two variables using SESOI to indicate practically significant difference** **A.** Scatterplot with SESOI depicted as grey band around linear regression line. **B.** Residual plot, where the difference between MAS and linear regression line (model estimate) is plotted against linear regression line (fitted or predicted MAS). SESOI is represented with the grey band. Residuals within SESOI band are of no practical difference. Dashed lines represent upper and lower *levels of agreement* using RSE and 95% confidence level (or in other words, 95% of the residuals distribution will be within these two dashed lines).

Magnitude-based estimators of the practically significant strength of the two variable association involve ratio between the SESOI ($SESOI_{upper} - SESOI_{lower}$) and RSE (SESOI to RSE), and PPER. SESOI to RSE indicates how big are the residuals compared to the SESOI, and thus a metric of the practical strength of the association. Assuming that residuals are being normally distributed, SESOI to RSE over 4 (or 2×1.96) would indicate excellent practical strength of the association. If you look at the Table 15, estimated SESOI to RSE in this example is not great, indicating poor practical strength of association.

Proportion of practically equivalent residuals (PPER) as a measure of the practical strength of the association revolves around estimating proportions of residuals in the *equivalent* range, defined as SESOI in the target variable (which is exactly the same as already introduced pEquivalent estimator). PPER can be estimated with the brute-force method by simply counting residuals in the equivalent zone,

Table 2.17: **Magnitude-based estimators of the association between two variables.** Association is estimated using linear regression model. MAS is the target variable, and YoYoIR1 is the predictor

SESOI YoYoIR1 (m)	SESOI MAS (km/h)	Sensitivity	RSE	SESOI MAS to RSE	PPER
± 40	± 0.5	0.11	0.22	4.57	0.98

or using algebraic method and assuming normally distributed residuals (i.e. using RSE of the residuals¹³).

Figure 2.11 graphically depicts how PPER is calculated. Practically significant association between two variables would have PPER equal to 1, which indicates that all residuals are within confines of the SESOI. If you look at the Table 2.17, estimated PPER in this example is almost perfect, indicating great practical strength of the association between YoYoIR1 and MAS tests.

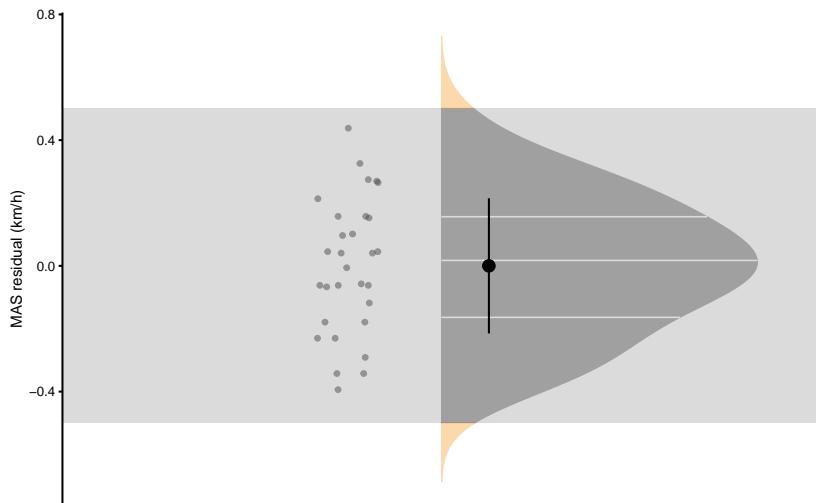


Figure 2.11: **Residuals of the linear regression model predicting MAS from YoYoIR1 test.** Proportion of residuals within SESOI band represent PPER

Visual inspection from the Figure 2.11 and magnitude-based estimates from the Table 2.17 indicate that using YoYoIR1 test scores, we are able to *predict*¹⁴ MAS test scores with the error within SESOI. But would that be the case if the we

¹³To estimate PPER algebraically, one can use residual SD, RSE, or RMSE since these are all measures of dispersion. In predictive models (see Prediction section) RMSE is utilized to estimate PPER.

¹⁴This is not ideal estimate of the predictive performance of this model as will be explained in the next section on [Prediction](#).

Table 2.18: **Magnitude-based estimators of the association between two variables.** Association is estimated using linear regression model. YoYoIR1 is the target variable, and MAS is the predictor

SESOI YoYoIR1 (m)	SESOI MAS (km/h)	Sensitivity	RSE	SESOI YoYoIR1 to RSE	PPER
± 40	± 0.5		6.56	133.84	0.6

want to predict YoYoIR1 from MAS test scores? Predictive performance of such model is depicted on the Figure 2.12 and magnitude-based estimator are enlisted in the Table 2.18.

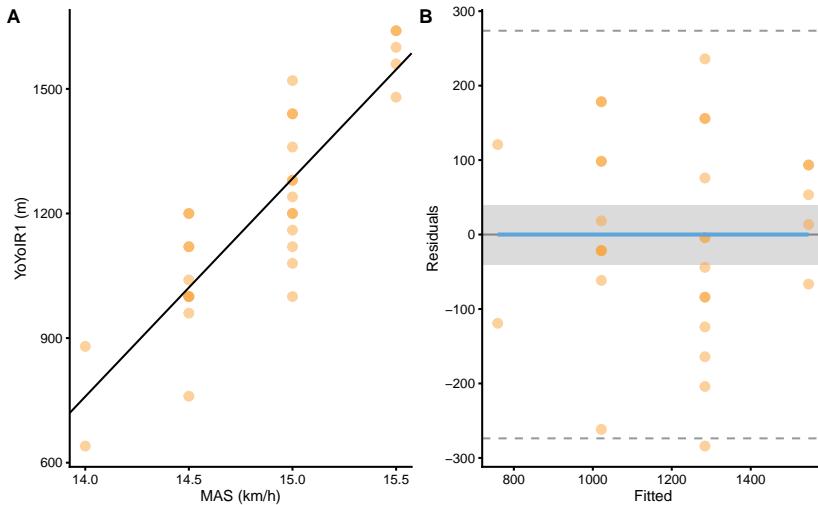


Figure 2.12: **Linear regression model estimating association between YoYoIR1 and MAS tests where YoYoIR1 is now the target variable.** **A.** Scatterplot with SESOI depicted as grey band around linear regression line. **B.** Residual plot, where the difference between YoYoIR1 and linear regression line (model estimate) is plotted against MAS variable. SESOI is represented with the grey band. Residuals within SESOI band are of no practical difference. Proportion of residuals within SESOI band represent PPER

As clearly indicated with this example, when estimating practical association between two variables, it is very important which variable is the target and which is predictor. When it comes to Pearson's r , R-Squared and MIC, this is not the case and results are same regardless of which variable is predictor and which is target.

From the analysis performed, it seems that predicting MAS from YoYoIR1 is practically useful and the association is practically significant. Unfortunately, the same is not the case when we try to predict YoYoIR1 from MAS. This might

be due different *physical traits* that determine the test scores. For example, results in the YoYoIR1 test might depend on the traits that include, but are not limited to, same traits important for the MAS test.

The purpose of descriptive analysis is only to describe - further analysis involving answering the *why* questions is in the domain of *explanatory modeling* and *causal inference* (which are covered in the [Causal inference](#) section), as well as [Advanced uses](#) of descriptive modeling, such as *latent variable modeling*. What is important to remember is that to describe magnitude-based association, it is important to clearly state which variable is the target and which is the predictor.

2.4 Advanced uses

Advanced techniques in the descriptive statistics involve dimension reduction, such as *principal component analysis* (PCA), latent variable modeling, such as *factor analysis* (FA), or cluster analysis (Beaujean 2014; Borsboom 2008; Borsboom, Mellenbergh, and van Heerden 2003; Everitt and Hothorn 2011; Finch and French 2015; Kabacoff 2015). These techniques are beyond the scope of this book and the interested readers are directed to references provided.

Chapter 3

Prediction

In many disciplines there is a near-exclusive use of the statistical models for causal inference¹ and the assumption that models with high explanatory power are inherently of high predictive power (Breiman 2001; Shmueli 2010; Yarkoni and Westfall 2017; Hernán, Hsu, and Healy 2019). There is a constant tug-of-war between prediction versus explanation, and experts are leaning on one side or the other. Some experts warn against over-reliance on explanatory models with poor predictive power (Breiman 2001; Shmueli 2010; Yarkoni and Westfall 2017), whereas some warn against over-reliance on predictive models that lack causal explanatory power that can guide intervention (Hernán, Hsu, and Healy 2019; Pearl and Mackenzie 2018; Pearl, Glymour, and Jewell 2016; Pearl 2019).

It is thus important to differentiate between the two and take into account the research question that we are trying to answer. In this book, I define predictive modeling by using definition from Galit Shmueli “as the process of applying a statistical model or data mining algorithm to data for the purpose of predicting new or future observations” (Shmueli 2010, 291). Usually this predictive statistical model is treated as a *black box*. Black box approach implies that we are not really interested in underlying mechanism and relationships between the predictor variables, only in predictive performance of the model (Breiman 2001; Shmueli 2010; Yarkoni and Westfall 2017)

Linear regression model from [Describing relationship between two variables](#) section already introduced predictive question (“If I know someone’s YoYoIR1 score, what would be his or her MAS score? Is the prediction within SESOI?”)

¹Some authors refer to causal inference as “explanatory” modeling (Breiman 2001; Shmueli 2010; Yarkoni and Westfall 2017), although Miguel Hernan warns against using such a somewhat misleading term “because causal effects may be quantified while remaining unexplained (randomized trials identify causal effects even if the causal mechanisms that explain them are unknown)” (Hernán, Hsu, and Healy 2019, 43). Andrew Gelman also makes distinctions between *forward causal inference* and *reverse causal inference* that might be useful in distinguishing between identifying causal effects and explaining them (Gelman 2011). This is elaborated in the [Causal inference](#) section of this book.

to complement the association one (“How is YoYoIR1 associated with MAS?”). This section will continue this quest and introduce essential concepts and caveats of the predictive analysis needed to answer predictive questions.

3.1 Overfitting

To explain a few caveats with predictive modeling, let’s take slightly more complex example (although we will come back to YoYoIR1 and MAS relationship later). Imagine we know the *true* relationship between back squat *relative 1RM* (BS)² and vertical jump height during a bodyweight squat jump (SJ; measured in cm). This true relationship is usually referred to as *data generating process* (DGP) (Carsey and Harden 2013) and one of the aims of causal inference tasks is to uncover parameters and mechanism of DGP from the acquired sample³. With predictive tasks this aim is of no direct interest, but rather reliable prediction regarding new or unseen observations.

DGP is usually unknown, but with simulations, such as this one, DGP is known and it is used to generate the sample data. Simulation is thus excellent teaching tool, since one can *play* with the problem and understand how the statistical analysis works, since the true DGP is known and can be compared with estimates (Carsey and Harden 2013; Hopkins 2007; Guillaume A Rousselet, Pernet, and Wilcox 2019a).

DGP is assumed to consist of *systematic component* $f(x)$ and *stochastic component* ϵ (Equation (3.1)).

$$Y = f(X) + \epsilon \quad (3.1)$$

Systematic component is assumed to be *fixed* in the population (constant from sample to sample) and captures the *true* relationship $f(X)$ among variables in the population (e.g. this can also be termed *signal*), while stochastic component represents *random noise* or *random error*, that varies from sample to sample, although its distribution remains the same. Random error is assumed to be normally distributed with mean of 0 and standard deviation which represents estimated parameter (either with RMSE or RSE). Thus, RMSE or RSE are *estimates* of ϵ .

In our example, the relationship between SJ and BS is expressed with the following Equation (3.2).

²For example, if an athlete lifted 175kg for a single rep in the back squat, and was unable to lift more, this represents his back squat 1RM, or one repetition maximum. Relative 1RM is calculated by dividing 1RM with athlete’s bodyweight. For example, an athlete with 175kg 1RM weights 85kg. His relative 1RM is equal to 2.05.

³Uncovering DGP parameters is not solely the goal of the causal inference (although causal inference task is to uncover or quantify causal mechanism), but also the main goal in the statistical inference where the aim is to quantify uncertainty about the *true* population parameters from the acquired sample. More about this topic in the [Statistical Inference](#) section.

$$SJ = 30 + 15 \times BS \times \sin(BS) + \epsilon \quad (3.2)$$

$$\epsilon \sim \mathcal{N}(0, 2)$$

Systematic component in the DGP is represented with $30 + 15 \times BS \times \sin(BS)$, and stochastic component is represented with the *known* random error (ϵ) that is normally distributed with the mean equal to zero and standard deviation equal to 2cm ($\mathcal{N}(0, 2)$). This random error can be termed *irreducible error* (James et al. 2017), since it is inherent to the true DGP. As will be demonstrated shortly, models that perform better than this irreducible error are said to *overfit*. In other words, models are jumping to the noise.

The objective of causal inference or explanatory modeling is to estimate the $f(X)$ (estimate is indicated with the *hat* symbol: $\hat{f}(x)$) or to understand the underlying DGP. With the predictive analysis, the goal is to find the best estimate of Y or \hat{y} . The underlying DGP is treated as a *black box*.

To demonstrate a concept of overfitting, we are going to generate two samples (N=35 observations) from the DGP with BS ranging from 0.8 to 2.5. These samples are *training* and *testing* sample (Figure 3.1). Training sample is used to *train* the prediction model, while *testing* sample will be used as a *holdout* sample for evaluating model performance on the *unseen* data.

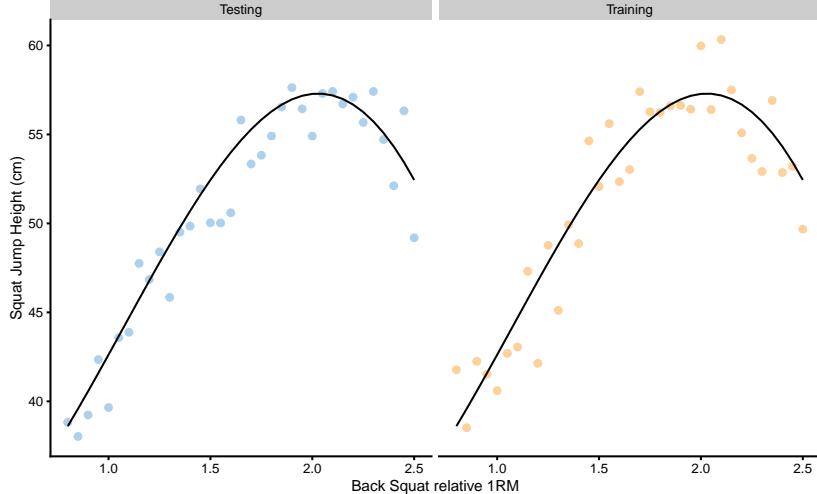


Figure 3.1: Two samples simulated from the known DGP. Black line represents systematic component of the DGP and it is equal for both training and testing samples. Observations vary in the two samples due stochastic component in the DGP

Model used to predict SJ from BS will be *polynomial linear regression*. Equation

(3.3) explains first, second, and third degree polynomial linear regression function and provides a form for n-degree polynomials. Please, note that first degree polynomial function represents simple linear regression.

$$\begin{aligned}
 \hat{y}_i &= \hat{\beta}_0 + \hat{\beta}_1 x_i^1 \\
 \hat{y}_i &= \hat{\beta}_0 + \hat{\beta}_1 x_i^1 + \hat{\beta}_2 x_i^2 \\
 \hat{y}_i &= \hat{\beta}_0 + \hat{\beta}_1 x_i^1 + \hat{\beta}_2 x_i^2 + \hat{\beta}_3 x_i^3 \\
 \hat{y}_i &= \hat{\beta}_0 + \hat{\beta}_1 x_i^1 + \cdots + \hat{\beta}_n x_i^n
 \end{aligned} \tag{3.3}$$

Increasing polynomial degrees increases the *flexibility* of the polynomial regression model, and thus can represent *tuning parameter* that we can select based on the model performance. In other words, we might be interested in finding polynomial degree that minimized model error (or maximize model fit). Figure 3.2 contains model performance on the training data for polynomial degrees ranging from 1 to 20.

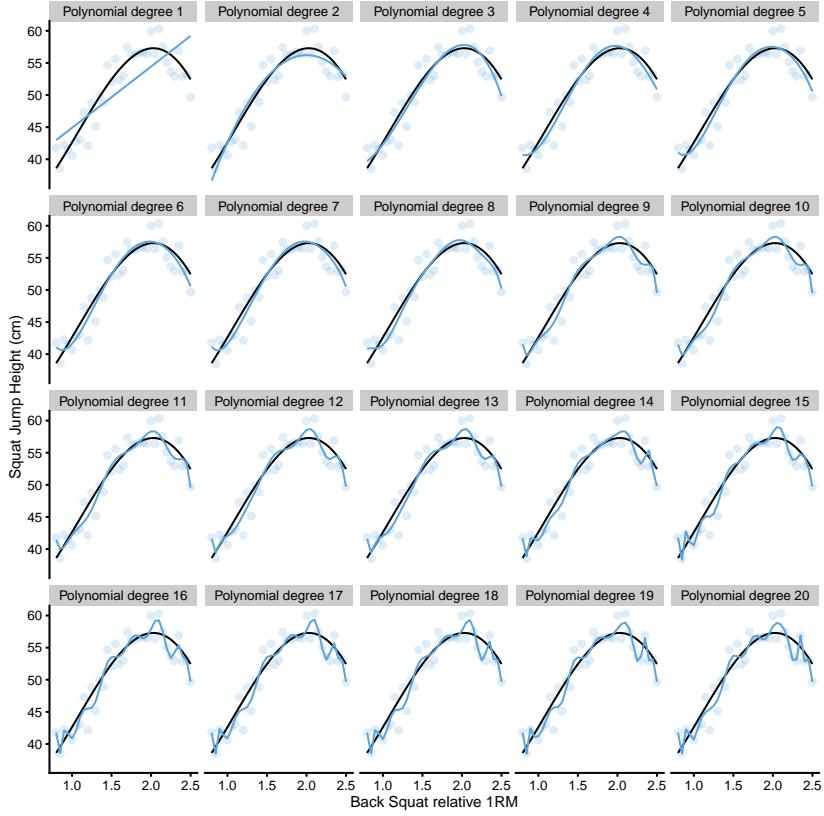


Figure 3.2: Model fit with varying polynomial degrees. More degrees equals better model fit

As can be seen from the Figure 3.2, the more flexible the model (or the higher the polynomial degree) the better it fits the data. But how do these models perform on the unseen, testing data sample? In order to quantify model performance, RMSE metric is used. Figure 3.3 demonstrates performance of the polynomial regression model on the training and testing data sample across different polynomial degrees.

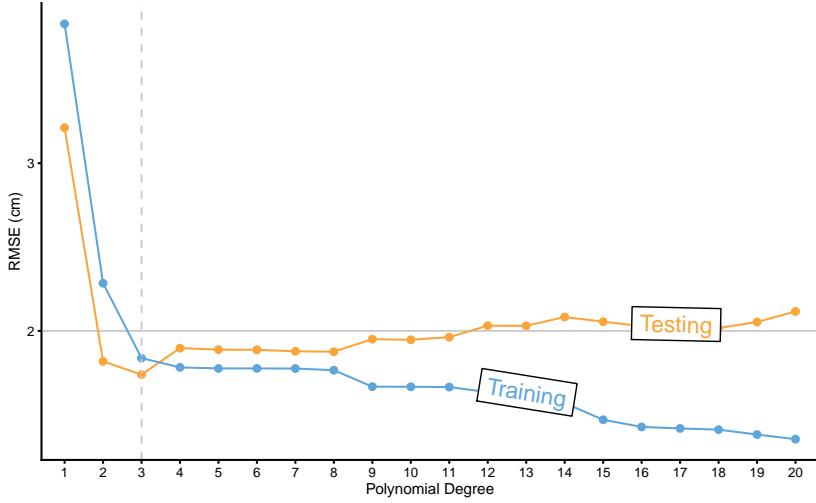


Figure 3.3: Testing and training errors across varying polynomial degrees. Model error is estimated with the RMSE metric, while polynomial degree represents tuning or flexibility parameter of the model. As can be noted from the figure, better training performance doesn't imply better testing performance. Vertical dashed line represents the polynomial degree at which testing error is lowest. Polynomial degrees on the right of the vertical dashed line are said to *overfit* the data, while polynomial degree on the left are said to *underfit* the data

As can be seen from the Figure 3.3, models with higher polynomial degrees tend to overfit (indicated by performance better than the known irreducible error ϵ visualized with the horizontal line at 2cm). Performance on the training data sample improves as the polynomial degrees increase, which is not the case with the performance on the testing data sample. There is clearly the best polynomial degree that has the best predictive performance on the unseen data. Polynomial degrees on the left of the vertical dashed line are said to *underfit*, while polynomial degrees on the right are said to *overfit*.

The take home message is that predictive performance on the training data can be too optimistic, and for evaluating predictive performance of the model, unseen data must be used, otherwise the model might overfit.

3.2 Cross-Validation

In order to evaluate predictive performance of the model, researchers usually remove some percent of data to be used as a testing or holdout sample. Unfortunately, this is not always possible (although it is recommended, particularly to evaluate final model performance, especially when there are multiple models

and model tuning). One solution to these problems is *cross-validation* technique (James et al. 2017; Kuhn and Johnson 2018; Yarkoni and Westfall 2017). There are numerous variations of the cross-validation, but the simplest one is *n-fold* cross validation (Figure 15). N-fold cross validation involve splitting the data into 5 to 10 equal folds and using one fold as a testing or hold-out sample while performing model training on the other folds. This is repeated over N-iteration (in this case 5 to 10) and the model performance is averaged to get *cross-validated model performance*.

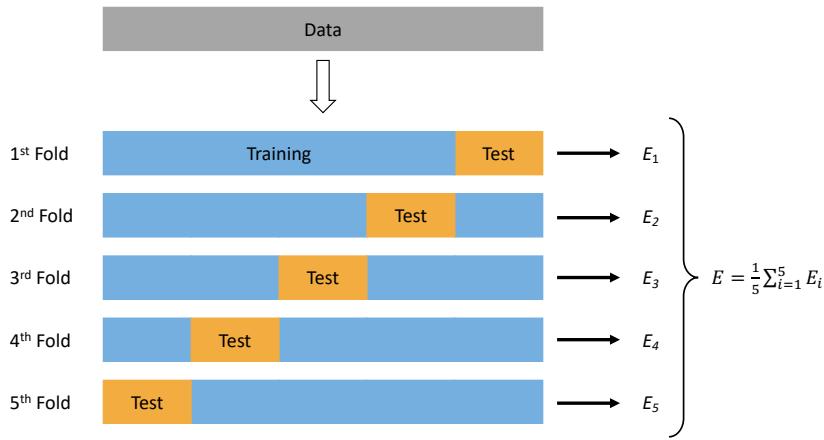


Figure 3.4: Cross-Validation

With predictive analysis and *machine learning*, different model's tuning parameters are evaluated (as well as multiple different models) to estimate the one that gives the best predictive performance. It is thus important to utilize techniques such as cross-validation to avoid overfitting and too optimistic model selection.

Certain models, such as *lasso*, *ridge regression*, and *elastic-net* implement *regularization* parameters that *penalizes* the model complexity and are used as a tuning variable (James et al. 2017; Kuhn and Johnson 2018; Yarkoni and Westfall 2017). This is useful in situations when there are a lot of predictors, and it is easy to overfit the model. Selecting the best regularization parameter that has the best cross-validated performance helps in simplifying the model and avoiding the overfit. These topics are beyond the scope of this book, and interested readers are directed to references provided.

3.2.1 Sample mean as the simplest predictive model

We have already discussed in [Sample mean as the simplest statistical model](#) section that sample **mean** can be considered simplest model that describes a particular sample with the lowest RMSE. But can it be used for prediction?

Table 3.1: Sample mean as prediction with associated prediction errors

Observed	Predicted	Error	Absolute Error	Squared Error
15	52.8	37.8	37.8	1428.84
19	52.8	33.8	33.8	1142.44
28	52.8	24.8	24.8	615.04
28	52.8	24.8	24.8	615.04
30	52.8	22.8	22.8	519.84
57	52.8	-4.2	4.2	17.64
71	52.8	-18.2	18.2	331.24
88	52.8	-35.2	35.2	1239.04
95	52.8	-42.2	42.2	1780.84
97	52.8	-44.2	44.2	1953.64

Here is an example to demonstrate both sample mean as a predictive model, as well as to demonstrate cross-validation technique. Let's assume that we have collected N=10 observations: 15, 19, 28, 28, 30, 57, 71, 88, 95, 97. Sample mean is equal to 52.8. If we assume that the sample mean represents our prediction for the observations, we can easily calculate *prediction error* for each observation, which is simple difference (column **Error** in the Table 3.1).

Besides simple difference, Table 3.1 provides errors (or losses) using two common *loss functions* (see [Sample mean as the simplest statistical model](#) section in [Description](#) chapter): *absolute loss* (column **Absolute Error**) and *quadratic loss* (column **Squared Error**).

We need to *aggregate* these errors or losses into a single metric using the *cost function*. If we calculate the mean of the prediction errors (column **Error** in the Table 3.1), we are going to get 0. This is because the positive and negative errors cancel each other out for the sample mean estimator. This error estimator is often referred to as *mean bias error* (MBE) or simply *bias* and is often used in *validity* and *reliability* analysis (see [Validity](#) and [Reliability](#) sections).

If we take the mean of the absolute prediction errors (column **Absolute error** in the Table 3.1) we are going to get *mean absolute error* (MAE) estimator, which is in this example equal to 28.8.

If we take the mean of the squared prediction errors (column **Squared error** in the Table 3.1) we are going to get *mean square error* (MSE) estimator, often called *variance* in the case of describing sample dispersion. In this example MSE is equal to 964.36. To bring back MSE to the same scale with the observation scale, square root of the MSE is taken. This represents *root mean square error* (RMSE), which is equal to 31.05. As explained in [Sample mean as the simplest statistical model](#) section, sample mean represents statistical model of the central tendency with the lowest RMSE.

Aforementioned error estimators, MBE, MAE, MSE, and RMSE can be considered different *cost functions*. Which one should be used⁴? As always, it depends (Chai and Draxler 2014). Assuming Gaussian normal distribution of the errors, MSE and RMSE have very useful mathematical properties that can be utilized in modeling stochastic or random components (i.e. *random error propagation* and prediction error decomposition in [Bias-Variance decomposition and trade-off](#) section). This property will be utilized thorough this book and particularly in the [Example of randomized control trial](#) section when estimating random or stochastic component of the treatment effect. I personally prefer to report multiple estimators, including error estimators, which is also a strategy suggested by Chai and Draxler (2014).

The are, of course, other loss and cost functions that could be used. For example, one might only use the *maximal error* (`MaxErr`) and *minimal error* (`MinErr`), rather than average. Discussion and review of these different metrics is beyond the scope of this book (for more information please check the package `Metrics` (Hamner and Frasco 2018) and the following references (Botchkarev 2019; Chai and Draxler 2014; Willmott and Matsuura 2005; Barron 2019)). Figure 3.5 visualize the most common loss functions that are used in both model training and as *performance metrics*. It is important to keep in mind that in the case of OLS regression, MSE (or RMSE) is minimized. It is thus important to make a distinction between cost function used in the optimization and model training (i.e. in OLS, parameters of the model are found so that MSE is minimized; in some machine-learning models `Huber loss` or `Ridge loss`⁵ is minimized; see Figure 3.5) versus cost function used as a performance metric (e.g. reporting `pEquivalent`, `MaxErr` or `MinErr` for OLS models).

⁴As explained in [Introduction](#) section, I will mostly focus at variables on continuous ratio scale. Prediction error metrics differ for target variable that is on nominal scale (i.e. *classification* task) and involve estimators such as `accuracy`, `specificity`, `sensitivity`, `area under curve` (AUC) (Kuhn and Johnson 2018; Lantz 2019).

⁵Although I've used the term *loss* here, these loss functions are also aggregated using `sum` or `mean` to create `Huber cost` or `Ridge cost`. `Huber loss` is a combination of absolute and quadratic losses and it's property is better *robustness* to outliers.

Table 3.2: Example cross-validation using sample mean as the prediction model

Fold	Training sample	mean	Testing Sample	MBE	MAE	RMSE	MinErr	MaxErr
1	15 19 28 -30 -71 88 95 -	49.43	- - - 28 - 57 - - - 97	-11.24	25.52	30.44	-47.57	21.43
2	15 19 -28 -57 -88 -97	50.67	- - 28 - 30 - 71 - 95 -	-5.33	27.00	28.81	-44.33	22.67
3	- - 28 28 30 57 71 - 95 97	58.00	15 19 - - - - 88 - -	17.33	37.33	37.73	-30.00	43.00

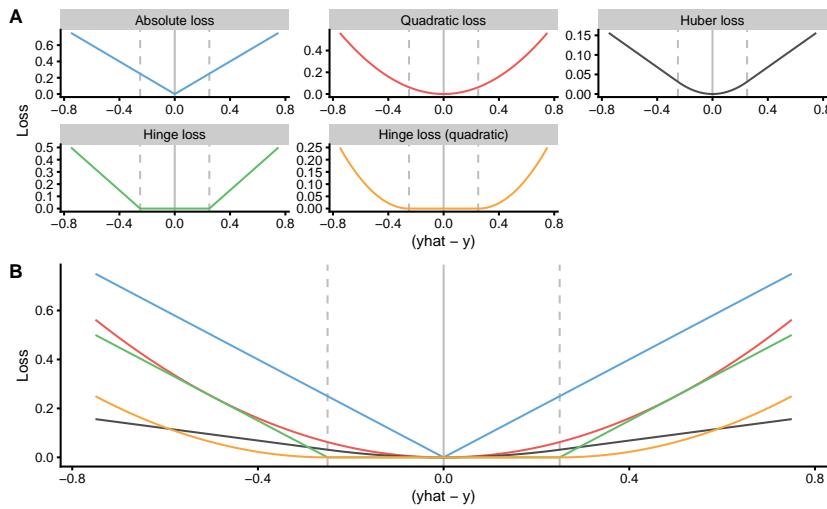


Figure 3.5: **Common loss functions.** **A.** Trellis plot of each loss function. **B.** Plot of all loss functions on a common scale. *Huber loss* is a combination of absolute and quadratic loss.

The take-away point is to understand that there are multiple performance metrics that can be utilized and it is best to report multiple of them.

Estimates for MBE, MAE, MSE⁶, and RMSE represent *training sample* predictive performance since sample `mean` is estimated using this very sample. We are interested how sample `mean`, as predictive model, perform on the unseen data. Cross-validation is one method to estimate model performance on unseen data. Table 3.2 contains 3-fold cross-validation sample used to train the model (in this case estimate sample `mean`) and to evaluate the performance on unseen data.

Since this is a small sample, we can repeat cross-validation few times. This is called *repeated cross-validation*. Let's repeat 3-folds cross-validation for 5 repeats (Table 3.3).

To calculate cross-validated prediction performance metrics, average of testing

⁶MSE will be removed from the further analysis and tables since it is equivalent to RMSE, just squared.

Table 3.3: Example repeated cross-validation using sample `mean` as the prediction model

Repeat	Fold	Training sample	mean	Testing Sample	MBE	MAE	RMSE	MinErr	MaxErr
1	1	15 19 28 -30 -71 88 95 -	49.43	-- 28 -57 -- -97	-11.24	25.52	30.44	-47.57	21.43
1	2	15 19 -28 -57 -88 -97	50.67	-- 28 -30 -71 -95 -	-5.33	27.00	28.81	-44.33	22.67
1	3	-- 28 28 30 57 71 -95 97	58.00	15 19 -- -88 --	17.33	37.33	37.73	-30.00	43.00
2	1	-19 28 -30 57 71 88 --	48.83	15 -- 28 -- -95 97	-9.92	37.25	38.83	-48.17	33.83
2	2	15 -28 28 -57 -- 95 97	53.33	-19 -- -30 -71 88 --	1.33	27.50	28.45	-34.67	34.33
2	3	15 19 -28 30 -71 88 95 97	55.38	-- 28 -- 57 ----	12.87	14.50	19.39	-1.63	27.37
3	1	-19 28 28 30 57 71 88 --	45.86	15 ----- 95 97	-23.14	43.71	44.66	-51.14	30.86
3	2	15 19 -30 -88 95 97	57.33	-- 28 28 -57 71 --	11.33	18.17	21.84	-13.67	29.33
3	3	15 -28 28 -57 71 -95 97	55.86	-19 -- -30 -88 --	10.19	31.62	31.94	-32.14	36.86
4	1	-19 28 -30 57 71 88 -97	55.71	15 -- 28 -- -95 -	9.71	35.90	36.37	-39.29	40.71
4	2	15 -28 30 -71 -95 97	56.00	-19 28 -- 57 -88 --	8.00	24.50	28.19	-32.00	37.00
4	3	15 19 28 28 -57 -88 95 -	47.14	-- -30 -71 -- -97	-18.86	30.29	33.41	-49.86	17.14
5	1	15 19 -28 -57 -88 95 97	57.00	-- 28 -30 -71 --	14.00	23.33	24.26	-14.00	29.00
5	2	15 -28 28 30 -71 88 95 -	50.71	-19 -- -57 -- -97	-6.95	28.10	32.60	-46.29	31.71
5	3	-19 28 -30 57 71 -97	50.33	15 -- 28 -- -88 95 -	-6.17	35.00	35.92	-44.67	35.33

Table 3.4: Cross-validated prediction performance metrics (estimators)

cvMBE	cvMAE	cvRMSE	cvMinErr	cvMaxErr
0.21	29.32	31.52	-35.29	31.37

MBE, MAE, RMSE, MinErr, and MaxErr is calculated and reported as cvMBE, cvMAE, cvRMSE, cvMinErr, and cvMaxErr (Table 3.4). Prediction performance metrics don't need to be averaged across cross-validation samples and can be instead estimated by *binding* (or *pooling*) all cross-validated samples together (i.e. target variable and predicted target variable). More about this at the end of this chapter in [Practical example: MAS and YoYoIR1 prediction](#) section.

As can be seen from the Table 3.4, all performance metrics estimated using repeated cross-validation are larger than the when estimated using the training data (full sample). Utilizing cross-validated estimates of performance (or error) should be used over training estimates when discussing predictive performance of the models. Unfortunately, this is almost never the case in sport science literature, where prediction is never estimated on unseen data and the model performance estimates can suffer from over-fitting.

Using sample `mean` as predictive model represents simplistic example, although this type of model is usually referred to as *baseline* model. Baseline models are used as benchmarks or anchor when discussing performance of more elaborate and complex predictive models. We will come back to these at the end of this section when we will perform predictive analysis of the linear regression model used in [Magnitude-based estimators](#) section for predicting MAS from YoYoIR1 (and *vice versa*) tests.

3.3 Bias-Variance decomposition and trade-off

Prediction error can be *decomposed* into two components, *reducible* and *irreducible* error⁷. Reducible error is the error that can be reduced with a better model, while irreducible error is the unknown error inherent to the DGP itself (James et al. 2017). Reducible error can be further divided into *Bias*² and *Variance* (Figure 3.6 and Equation (3.4)).

$$\begin{aligned} \text{Prediction error} &= \text{Reducible error} + \text{Irreducible error} \\ \text{Prediction error} &= (\text{Bias}^2 + \text{Variance}) + \text{Irreducible error} \end{aligned} \quad (3.4)$$

*Bias*² represents constant or systematic error, which is introduced by approximating a real-life problem, which may be extremely complicated, by a much simpler model (James et al. 2017). *Variance* represents variable or random error, and refers to the amount by which model parameters would change if we estimated it by using a different training data set (James et al. 2017).

⁷As it will be explained in [Statistical inference](#) section, there are two kinds of uncertainty: *aleatory* and *epistemic*. Aleatory uncertainty is inherent randomness and it is usually represented as irreducible error. Epistemic uncertainty is due to the lack of knowledge or information, which can be considered reducible error. In other words, better models or models with more information will be able to reduce prediction error by reducing the reducible or epistemic uncertainty. The upper ceiling of the predictive performance is determined by irreducible error (which is unknown). Please refer to the [Statistical inference](#) section for more information.

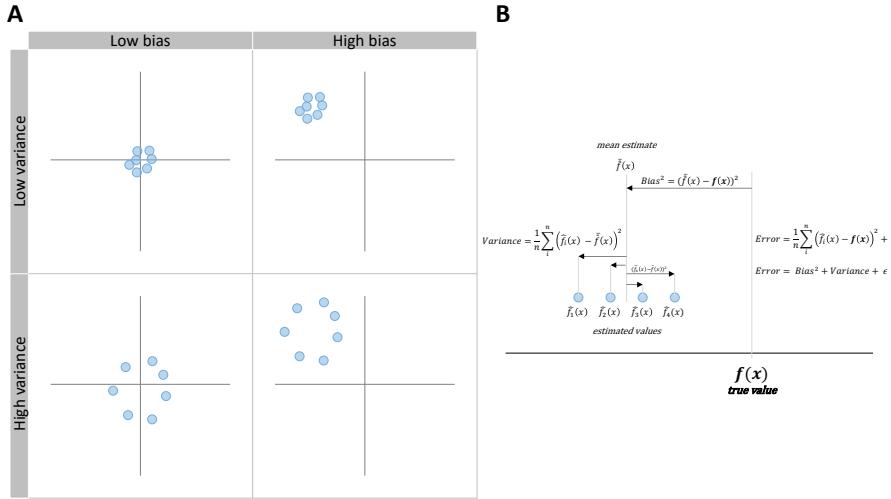


Figure 3.6: **Bias and variance decomposition of the error.** **A.** Visual representation of $Bias^2$ and $Variance$ using the shooting target. **B.** Error can be decomposed to $Bias^2$, $Variance$, and $Irreducible\ error$, where $Bias^2$ represents constant or systematic error and $Variance$ represents variable or random error

To understand this concept, we need to run a simulation using *known* relationship between BS and SJ (see Figure 3.1, Figure 3.2, and Equation (3.2)). Prediction error decomposition to bias and variance is done for a *single* data point. To do that we need to differentiate between the following variables:

- x predictors. In our case we only have one x predictor - BS and we will use $BS = 1.75$ for this simulation
- y_{true} value for a particular x values. In our case SJ variable represents target variable y , which is equal to $SJ = 30 + 15 \times BS \times \sin(BS)$ or $SJ = 55.83\text{cm}$. Both x and y_{true} values are constant across simulations
- $y_{observed}$ represents observed y which differs from y_{true} due stochastic component ϵ . This implies that $y_{observed}$ randomly varies across simulations. The equation for $y_{observed}$ is: $SJ = 30 + 15 \times BS \times \sin(BS) + \mathcal{N}(0, 2)$. Error ϵ represents irreducible error, because it is inherent to DGP and it is equal to $\mathcal{N}(0, 2)$.
- $y_{predicted}$ represents model prediction using the training sample of x and $y_{observed}$ values.

For every simulation ($N=200$ simulations in total), the whole sample of $N=35$ observations is generated from the known DGP. This includes x , y_{true} , and $y_{observed}$ variables. Polynomial regression models (from 1 to 20 polynomial degrees) are being fitted (or trained) using x predictors (in our case BS variable)

Table 3.5: Results of first 10 simulations for 2nd degree polynomial linear regression model

sim	model	x	y_true	y_observed	y_predicted
1	2	1.75	55.83	54.65	55.27
2	2	1.75	55.83	53.80	55.34
3	2	1.75	55.83	56.03	55.34
4	2	1.75	55.83	55.71	55.68
5	2	1.75	55.83	55.52	54.91
6	2	1.75	55.83	57.55	55.81
7	2	1.75	55.83	52.03	55.08
8	2	1.75	55.83	56.35	55.92
9	2	1.75	55.83	54.05	54.64
10	2	1.75	55.83	58.16	55.57

and $y_{observed}$ as our target variable. True y (y_{true}) is thus unknown to the model, but only to us. After models are trained, we estimate $y_{predicted}$ using $BS = 1.75$, for which the y_{true} is equal to $SJ = 55.83\text{cm}$.

Table 3.5 contains results of the first 10 simulations for 2nd degree polynomial model.

To estimate reducible error, MSE estimator is used (Equation (3.5)).

$$\text{Reducible error} = \frac{1}{n_{sim}} \sum_{j=1}^{n_{sim}} (y_{predicted,j,x=1.75} - y_{true,x=1.75})^2 \quad (3.5)$$

Reducible error can be decomposed to $Bias^2$ and $Variance$, or systematic error and variable or random error. $Bias^2$ is squared difference between y_{true} and mean of $y_{predicted}$ across simulations (Equation (3.6)).

$$Bias^2 = \left(\frac{1}{n_{sim}} \sum_{j=1}^{n_{sim}} (y_{predicted,j,x=1.75}) - y_{true,x=1.75} \right)^2 \quad (3.6)$$

$Variance$ represents, pretty much, SD of the of the $y_{predicted}$ and it is an estimate of how much the predictions vary across simulations (Equation (3.7)).

$$Variance = \frac{1}{n_{sim}} \sum_{j=1}^{n_{sim}} (y_{predicted,x=1.75} - \bar{y}_{predicted,x=1.75})^2 \quad (3.7)$$

Reducible error is thus equal to the sum of $Bias^2$ and $Variance$.

If you remember from Description section, $Bias^2$ and $Variance$ are nothing more than measure of central tendency (i.e. systematic or constant error) and

Table 3.6: Calculated errors for all 200 simulations for 2nd degree polynomial linear regression

model	x	y_true	Prediction error	Bias ²	Variance	Irreducible error
2	1.75	55.83	5.17	0.09	0.21	4.87

measure of spread (i.e. variable or random error). Figure 3.6 also illustrates this concept.

Irreducible error is estimated using **MSE** as well (Equation MSE-irreducible-error).

$$\text{Irreducible error} = \frac{1}{n_{sim}} \sum_{j=1}^{n_{sim}} (y_{\text{observed}_{x=1.75}} - y_{\text{true}_{x=1.75}})^2 \quad (3.8)$$

Since we know that the stochastic error in the DGP is normally distributed with SD=2cm, expected irreducible error should be around 4cm (this is because **MSE** is mean squared error, and **RMSE**, which is equivalent to **SD**, is calculated by doing square root of **MSE**). This might not be exactly 4cm due *sampling error* which is the topic of [Statistical inference](#) section.

As explained in Equation (3.4), *Prediction error* is equal to sum of *Bias²*, *Variance* and *Irreducible error*. Table contains estimated aforementioned errors using all 200 simulations for 2nd degree polynomial linear regression.

This decomposition of errors is one useful mathematical property when using squared errors that I alluded to in the [Cross-Validation](#) section when discussing prediction error metrics.

If we perform this analysis for each degree of polynomial fit, we will estimate prediction error, as well as *Bias²* and *Variance* across model complexity (i.e. polynomial degrees). This is visualized in the Figure 3.7.

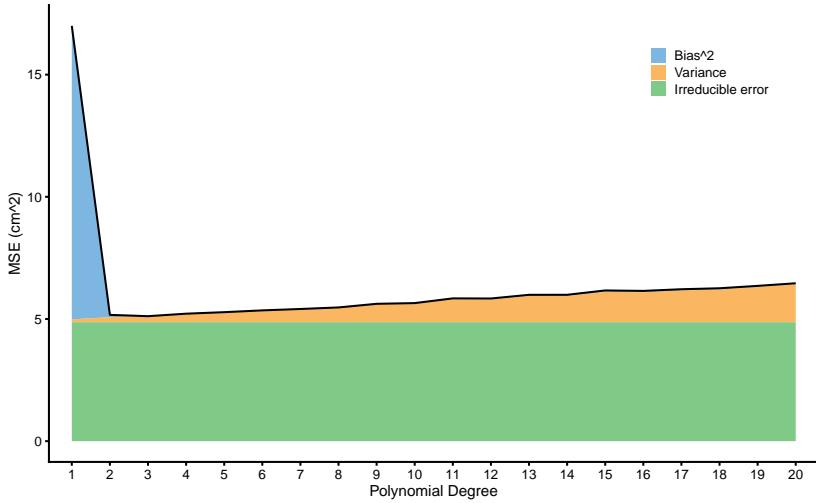


Figure 3.7: **Bias and Variance error decomposition.** *Prediction error* is indicated with the black line, and is decomposed to *Bias*², *Variance*, and *Irreducible error*. These are represented with areas of different color

Beside decomposition of *Prediction error* to *Bias*², *Variance* and *Irreducible error*, it is important to notice the *trade-off* between *Bias*² and *Variance*. Linear regression models with lower polynomial degree, particularly 1st degree which is simple linear regression, has higher *Bias*² due imposed *linearity* of the model (we can say that linear regression model is more *biased*). As *Bias*² decreases with more flexible models (i.e. higher polynomial degree), *Variance* increase due model being too *jumpy* across simulations. To achieve best prediction (or the lower *Prediction error*) a balance between *Bias*² and *Variance* needs to be found, both within a particular model and across models. The *free lunch* theorem (Kuhn and Johnson 2018; Yarkoni and Westfall 2017) states that there is no single model that is the best across all different sets of problems. One needs to evaluate multiple models⁸ to estimate which one is the best for a particular problem at hand.

To estimate *Bias*² and *Variance*, *true DGP* must be known. Unfortunately, we do not know these for the real world problems, only for simulations. But even when we do not know y_{true} values (and thus *Irreducible error*), concepts of *Bias*² and *Variance* can be applied in cross-validation samples (particularly when using multiple repeats) and can be estimated using $y_{observed}$. I will provide one such analysis in [Practical example: MAS and YoYoIR1 prediction section](#).

⁸Or to utilize subject matter knowledge needed to select the model. More about this topic can be found in the [Subject matter knowledge](#) section.

3.4 Interpretability

As explained, predictive models put predictive performance over explanation of the underlying DGP mechanism (which is treated as a black box). However, sometimes we might be interested in which predictor is the most important, how do predictions change when particular predictor changes, or why⁹ model made a particular prediction for a case of interest (Kuhn and Johnson 2018; Molnar 2018; Ribeiro, Singh, and Guestrin 2016). Model interpretability can be defined as the degree to which a human can understand the cause of a decision (Miller 2017; Molnar 2018; Biecek and Burzykowski 2019). Some models are more inherently interpretable (e.g. linear regression) and some are indeed very complex and hard to interpret (e.g. random forest or neural networks). For this reason, there are *model-agnostic* techniques that can help increase model interpretability.

Excellent book and R (R Core Team 2020) package by Christoph Molnar (Molnar, Bischl, and Casalicchio 2018; Molnar 2018) demonstrates a few model-agnostic interpretation techniques. One such technique is estimating which predictor is the most important (*variable importance*). One method for estimating variable importance involves *perturbing* one predictor and estimating the change in model performance. The predictor whose perturbing causes the biggest change in model performance can be considered the most important (Kuhn and Johnson 2018; Molnar 2018). There are others approaches for estimating variable importance (B. M. Greenwell 2017b). Those interested in further details can also check *vip* (Greenwell, Boehmke, and Gray 2020) R (R Core Team 2020) package.

One might be interested in how the predicted outcome changes when particular predictor changes. Techniques such as *partial dependence plot* (PDP), *individual conditional expectation* (ICE) and *accumulated local effects* (ALE) can be helpful in interpreting the effect of particular predictor on predicted outcome (Molnar, Bischl, and Casalicchio 2018; Molnar 2018; Goldstein et al. 2013; Zhao and Hastie 2019; B. M. Greenwell 2017a). Similar techniques are utilized in [Prediction as a complement to causal inference](#) section. Interested readers are also directed toward *visreg* (Breheny and Burchett 2017) and *effects* (Fox and Weisberg 2018; Fox 2003; Fox and Hong 2009) R (R Core Team 2020) packages for more information about visualizing models.

⁹With the recent laws such as European’s DGPR, predictive models needs to be able to explain or provide explanation why particular decision or prediction has been made. Christoph Molnar explains the need for model interpretability with one interesting example: “By default, machine learning models pick up biases from the training data. This can turn your machine learning models into racists that discriminate against protected groups. Interpretability is a useful debugging tool for detecting bias in machine learning models. It might happen that the machine learning model’s, you have trained for, automatic approval or rejection of credit applications discriminates against a minority. Your main goal is to grant loans only to people who will eventually repay them. The incompleteness of the problem formulation in this case lies in the fact that you not only want to minimize loan defaults, but are also obliged not to discriminate on the basis of certain demographics. This is an additional constraint that is part of your problem formulation (granting loans in a low-risk and compliant way) that is not covered by the loss function the machine learning model was optimized for.” (Molnar 2018, 11)

It is important to keep in mind that these model-agnostic explanations should not be automatically treated as causal explanations (Pearl and Mackenzie 2018; Pearl 2019), but as mere association and descriptive analysis that can still be useful in understanding and interpreting the underlying predictive *black-box*. They are not without problems, such as correlated variables, interactions and other issues (Altmann et al. 2019).

According to Judea Pearl (Pearl and Mackenzie 2018; Pearl 2019), prediction models should belong to the first level of *ladder of causation*¹⁰, which represents simple “curve fitting”. Although in under special conditions these techniques can have causal interpretation (Zhao and Hastie 2019).

The distinctions, similarities and issues between predictive modeling, machine learning and causal inference is currently hot topic in debates between machine learning specialists, statisticians and philosophers of science and it is beyond the scope of this book to delve into the debate. Interested readers are directed towards work by Miguel Hernan (Hernán 2017, 2016, 2018; Hernán and Robins, n.d.; Hernán, Hsu, and Healy 2019), Judea Pearl (Pearl and Mackenzie 2018; Pearl, Glymour, and Jewell 2016; Pearl 2019, 2009), Samantha Kleinberg (Kleinberg 2015, 2018) and others (Watts et al. 2018; Saddiki and Balzer 2018; Kleinberg, Liang, and Mullainathan 2017; Breiman 2001; Shmueli 2010; Yarkoni and Westfall 2017). The next [Causal Inference](#) section introduces the causal inference as a specific task of statistical modeling.

3.5 Magnitude-based prediction estimators

Similar to the magnitude-based estimators from [Describing relationship between two variables](#) section, one can utilize target variable SESOI to get magnitude-based estimates of predictive performance of the model. Rather than utilizing RSE as an estimate of the model fit in the training data, one can utilize *cross-validated RMSE* (cvRMSE), *SESOI to cvRMSE*, as well as *cross-validated proportion of practically equivalent residuals* (cvPPER) estimators.

Continuing with the squat jump height and relative squat 1RM example, one can assume that the SESOI in the squat jump is $\pm 1\text{cm}$. For the sake of example, we can *feature engineer* (Kuhn and Johnson 2018, 2019) relative squat 1RM variable to include all 20 degree polynomials. This way, we have created 20 predictor variables. To avoid overfitting, *elastic-net model* (Friedman, Hastie, and Tibshirani 2010) implemented in the *caret* R package (Kuhn and Johnson 2018;

¹⁰“Pearl’s causal meta-model involves a three-level abstraction he calls the ladder of causation. The lowest level, *Association* (seeing/observing), entails the sensing of regularities or patterns in the input data, expressed as correlations. The middle level, *Intervention* (doing), predicts the effects of deliberate actions, expressed as causal relationships. The highest level, *Counterfactuals* (imagining), involves constructing a theory of (part of) the world that explains why specific actions have specific effects and what happens in the absence of such actions” (Wikipedia contributors 2019)

Table 3.7: **Common predictive metrics and magnitude-based predictive metrics.** Metrics starting with **cv** indicate cross-validated performance metrics. Metrics without **cv** indicate performance metrics on the training data set, which is often more optimistic

SESOI (cm)	cvRMSE (cm)	SESOI to cvRMSE	cvPPER	RMSE (cm)	SESOI to RMSE	PPER
±1	2.19	0.91	0.33	1.99	1.01	0.38

Kuhn et al. 2018) is utilized, as well as repeated cross-validation involving 3 splits repeated 10 times. Predictive model performance is evaluated by using **cvRMSE**, together with magnitude-based performance estimators (**SESOI to cvRMSE** and **cvPPER**).

Elastic-net model represents regression method that linearly combines the $L1$ and $L2$ penalties of the lasso and ridge methods, or *alpha* and *lambda* tuning parameters (Friedman, Hastie, and Tibshirani 2010; James et al. 2017; Kuhn and Johnson 2018). Total of nine combinations of tuning parameters is evaluated using aforementioned repeated cross-validation, and the model with minimal **cvRMSE** is selected as the best one. Performance metrics of the best model are further reported. Table 3.7 contains cross-validated best model performance metrics together with model performance on the training data set.

Utilizing *a priori* known SESOI gives us practical *anchor* to evaluate predictive model performance. Reported **SESOI to cvRMSE** (0.91) as well as **cvPPER** (0.33) indicate very poor predictive performance of the model. In practical terms, utilizing relative squat 1RM doesn't produce practically meaningful predictions given SESOI of ±1cm and the model as well as the data sample utilized.

Model performance can be visualized using the training data set (Figure 3.8). PPER estimator, for both cross-validate estimate and training data performance estimate, utilized SD of the residuals and provided SESOI. Grey band on panels A and B on Figure 3.8 represents SESOI, and as can be visually inspected, model residuals are much wider than the SESOI, indicating poor practical predictive performance.

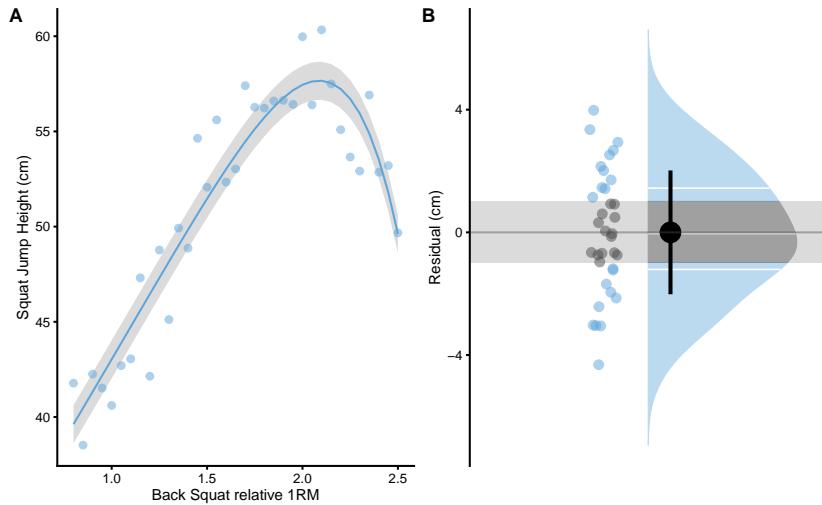


Figure 3.8: Model performance on the training data set. A. Model with the lowest cvRMSE is selected. SESOI is depicted as grey band around the model prediction (blue line). **B.** Residuals scatter plot. Residuals outside of the SESOI band (grey band) indicate prediction which error is practically significant. PPER represents proportion of residuals inside the SESOI band

Predictive tasks are focusing on providing the best predictions on the novel or unseen data without much concern about the underlying DGP. Predictive model performance can be evaluated by using magnitude-based approach to give insights into practical significance of the predictions. These magnitude-based prediction estimators, can be used to complement explanatory or causal inference tasks, rather than relying solely on the group-based and average-based estimators. This topic is further discussed in the [Prediction as a complement to causal inference](#) section.

3.6 Practical example: MAS and YoYoIR1 prediction

In [Describing relationship between two variables](#) we have used two physical performance tests, MAS and YoYoIR1, to showcase relationship or association between two variables. Besides mere association, we have stepped into the domain of prediction by utilizing magnitude-based estimators such as PPER and SESOI to RMSE. As you have learned so far in this section, these predictions were made on the training data set. Let's implement concepts learned so far to estimate predictive performance on the unseen data. Why is this important? Although very simple model, we are interested in predicting MAS from YoYoIR1

test score for a new or unseen individual. For this reason, it is important to get estimates of model performance on the unseen athletes.

3.6.1 Predicting MAS from YoYoIR1

Let's first estimate predictive performance when predicting MAS scores from the YoYoIR1 score using MAS SESOI $\pm 0.5\text{km/h}$ and linear regression.

Figure 3.9 consists of two panels. Panel A depicts scatter plot between YoYoIR1 and MAS scores (black line represents linear model fit). Panel B depicts $y_{predicted}$ (predicted or fitted MAS using simple linear regression; i.e. the black line on the panel A) against the model residuals $y_{residual} = y_{predicted} - y_{observed}$, or Predicted MAS - MAS. The data points represent model performance on the full training data set.

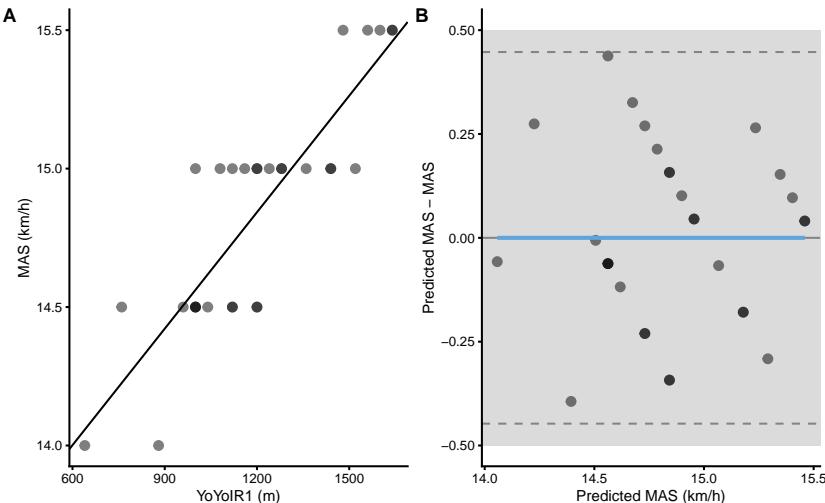


Figure 3.9: Scatter plot for simple linear regression between MAS and YoYoIR1 using the full training data sample. A. Scatter plot between MAS and YoYoIR1 scores. Black line indicates model prediction. **B.** Scatter plot between $y_{predicted}$ (fitted or predicted MAS) against model residual $y_{residual} = y_{predicted} - y_{observed}$, or Predicted MAS - MAS. Dotted lines indicate *Levels of Agreement* (LOA; i.e. upper and lower threshold that contain 95% of residuals distribution) and grey band indicates SESOI. Blue line indicate linear regression fit of the residuals and is used to indicate issues with the model (residuals)

Predictive performance for the full training data set is enlisted in the Table 3.8.

But as already explained, these are not predictive performance estimators for the unseen data. To estimate how the model performs on the unseen data

Table 3.8: Predictive performance using the full training data set

MBE	MAE	RMSE	PPER	SESOI.to.RMSE	R.squared	MinErr	MaxErr	MaxAbsErr	
0	0.17	0.21	0.97		4.73	0.74	-0.44	0.39	0.44

Table 3.9: Predictive performance for every repeated cross-validated sample

fold	MBE	MAE	RMSE	PPER	SESOI to RMSE	R-squared	MinErr	MaxErr	MaxAbsErr	
1	Fold1.Rep1	-0.13	0.27	0.29	0.87	3.42	0.62	-0.46	0.38	0.46
2	Fold1.Rep2	-0.04	0.13	0.14	0.99	7.15	0.87	-0.22	0.15	0.22
3	Fold1.Rep3	0.13	0.24	0.27	0.90	3.74	0.79	-0.27	0.49	0.49
4	Fold1.Rep4	-0.17	0.25	0.28	0.89	3.52	0.41	-0.51	0.26	0.51
5	Fold1.Rep5	0.08	0.20	0.24	0.92	4.20	0.77	-0.28	0.46	0.46
6	Fold2.Rep1	0.09	0.15	0.19	0.97	5.18	0.87	-0.27	0.37	0.37
7	Fold2.Rep2	0.05	0.20	0.24	0.93	4.16	0.71	-0.31	0.41	0.41
8	Fold2.Rep3	-0.14	0.18	0.22	0.96	4.65	0.81	-0.38	0.18	0.38
9	Fold2.Rep4	-0.02	0.13	0.17	0.98	5.92	0.75	-0.30	0.33	0.33
10	Fold2.Rep5	-0.07	0.21	0.27	0.89	3.76	0.27	-0.50	0.33	0.50
11	Fold3.Rep1	0.04	0.13	0.16	0.99	6.27	0.60	-0.20	0.31	0.31
12	Fold3.Rep2	-0.01	0.20	0.24	0.92	4.21	0.54	-0.45	0.34	0.45
13	Fold3.Rep3	0.02	0.15	0.21	0.95	4.74	0.51	-0.44	0.35	0.44
14	Fold3.Rep4	0.21	0.23	0.28	0.91	3.57	0.83	-0.06	0.53	0.53
15	Fold3.Rep5	-0.01	0.17	0.19	0.97	5.29	0.82	-0.32	0.34	0.34

(i.e. unseen athletes in this case), cross-validation is performed using 3 folds and 5 repeats. Estimated predictive performance for every cross-validation sample is enlisted in the Table 3.9.

As explained in [Cross-Validation](#) section, to calculate overall cross-validated performance, the `mean` is calculated for the performance metrics in the Table 3.9. Besides reporting the `mean` as the summary for predictive performances across cross-validated samples, `SD`, `min`, and `max` can be reported too. Another method of summarizing predictive performance over cross-validated samples would be to *bind* or *pool* all $y_{observed}$ and $y_{predicted}$ scores from the test samples together and then calculate *overall* predictive performance metrics. These pooled cross_valuated $y_{observed}$ and $y_{predicted}$ can also be visualized using the residuals plot (Panel C in Figure 3.10).

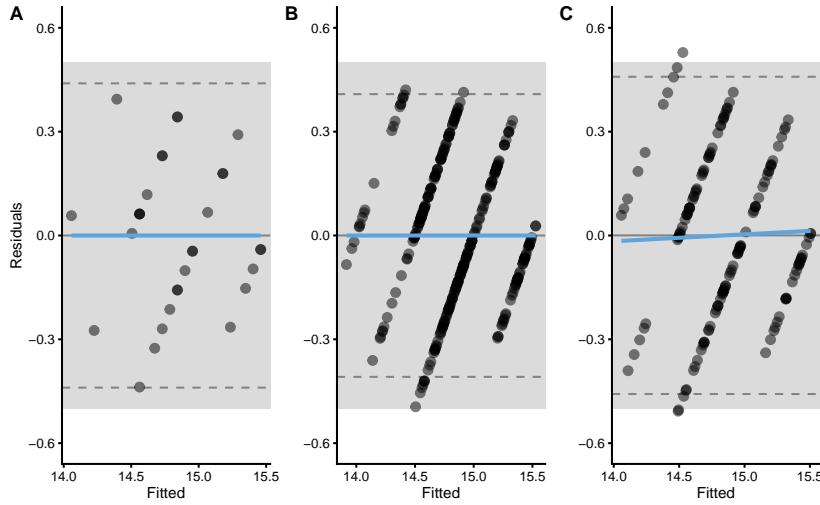


Figure 3.10: **Residuals plot.** **A.** Model residuals using the training data. This is exactly the same as panel B in Figure 3.9. **B.** Model residuals using the cross-validated training data. **C.** Model residuals using the cross-validated testing data.

As can be seen from the panels B and C in Figure 3.10, since we have used 5 repeats of the 3-fold cross-validations, each $y_{observed}$ will have 5 assigned $y_{predicted}$ scores. These can be used to estimate $Bias^2$ and $Variance$ as explained in the [Bias-Variance decomposition and trade-off](#) section. Since we do not know the y_{true} scores, we can only estimate $Bias^2$ and $Variance$ of the model for each $y_{observed}$ using cross-validated $y_{predicted}$. This is, of course, only possible if multiple repeats of cross-validation are performed. It bears repeating that this $Bias^2$ and $Variance$ decomposition of the prediction error using cross-validated $y_{predicted}$ and $y_{observed}$ are **NOT** the same as when using simulation and known DGP as done in [Bias-Variance decomposition and trade-off](#) section.

But these can be useful diagnostic tools for checking where the model fails (e.g. what particular observation might be problematic or outlier, as well how does $Bias^2$ and $Variance$ changes across $y_{observed}$ continuum). These two concepts are depicted on Figure 3.11 and Figure 3.12.

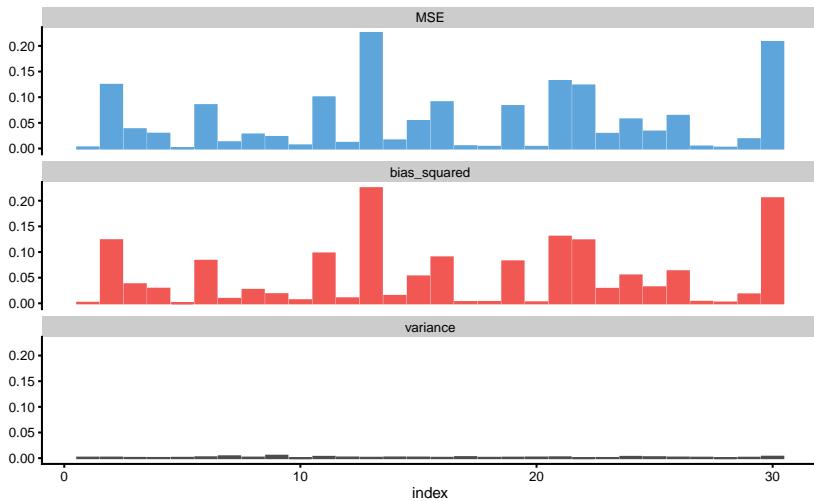


Figure 3.11: **Prediction error (MSE), $Bias^2$, and Variance across repeated cross-validated testing data.** X-axis on the panels represents observation $index$, as in $y_{i, \text{observed}}$

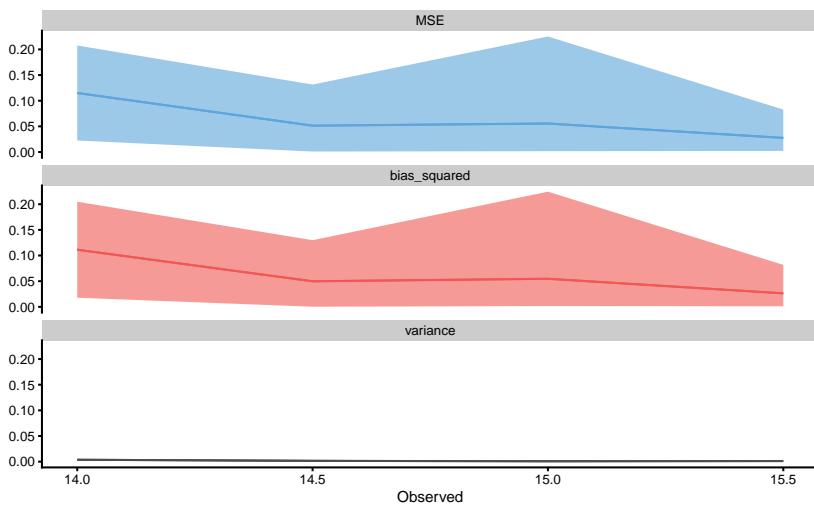


Figure 3.12: **Prediction error (MSE), $Bias^2$ and Variance across repeated cross-validated testing data.** X-axis on the panels represent y_{observed} . Since there might be multiple equal y_{observed} , **min** and **max** are used and represent **ribbon over mean** (indicated by line)

Since $Bias$ and $Variance$ represent a quantitative summary of the residuals

across cross-validations, the residuals and predicted observations across cross-validation testing folds can be visualized in more details as depicted on Figures 3.13 and 3.14.

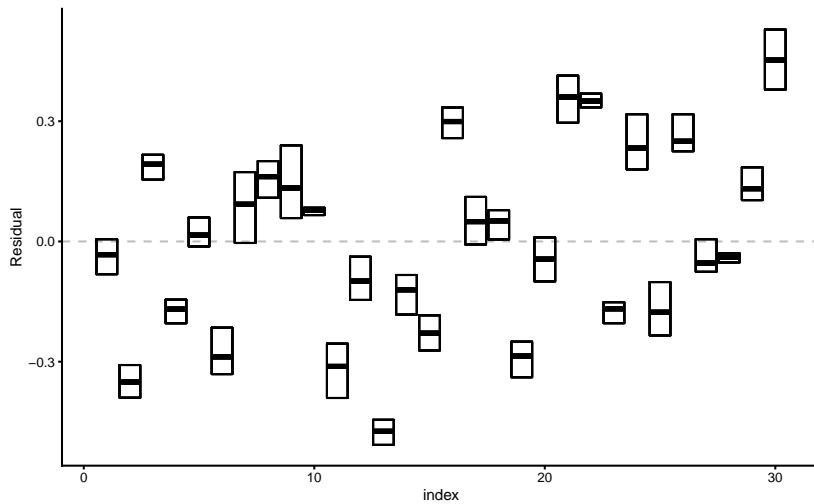


Figure 3.13: Testing prediction residuals across cross-validation folds summarized with cross-bars for every observation. Cross-bars represent ranges of testing residuals for each observation, while horizontal bar represent mean residual. The length of the bar represents *Variance*, while distance between horizontal dashed line and horizontal line in the cross-bar (i.e. mean residual) represents *Bias*.

Table 3.10: Predictive performance summary

metric	training	training.pooled	testing.pooled	mean	SD	min	max
MBE	0.00	0.00	0.00	0.00	0.10	-0.17	0.21
MAE	0.17	0.17	0.19	0.19	0.05	0.13	0.27
RMSE	0.21	0.21	0.23	0.23	0.05	0.14	0.29
PPER	0.97	0.98	0.97	0.94	0.04	0.87	0.99
SESOI to RMSE	4.73	4.83	4.33	4.65	1.12	3.42	7.15
R-squared	0.74	0.75	0.68	0.68	0.18	0.27	0.87
MinErr	-0.44	-0.49	-0.51	-0.33	0.13	-0.51	-0.06
MaxErr	0.39	0.42	0.53	0.35	0.10	0.15	0.53
MaxAbsErr	0.44	0.49	0.53	0.41	0.09	0.22	0.53

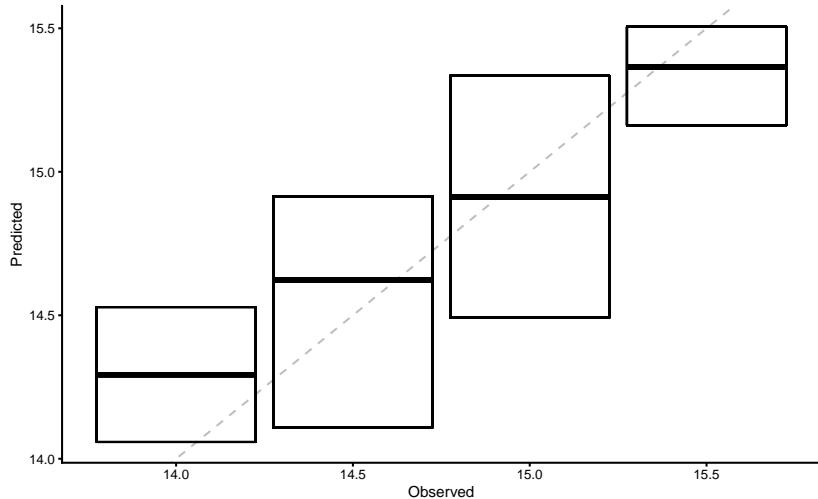


Figure 3.14: **Testing prediction residuals across cross-validation folds summarized with cross-bars for every observation value.** Cross-bars represent ranges of testing residuals for each observation, while horizontal bar represent mean residual. The length of the bar represents *Variance*, while distance between horizontal dashed line and horizontal line in the cross-bar (i.e. mean residual) represents *Bias*.

Cross-validated, *pooled*, and full training data set predictive performance metrics can be found in the Table 3.10. Please note that the *pooled* predictive performance metrics are in the column `testing.pooled`.

Summary from the Table 3.10 as well as the individual cross-validated sample predictive performance from the Table 3.9 are visually represented in the Figure 3.15).

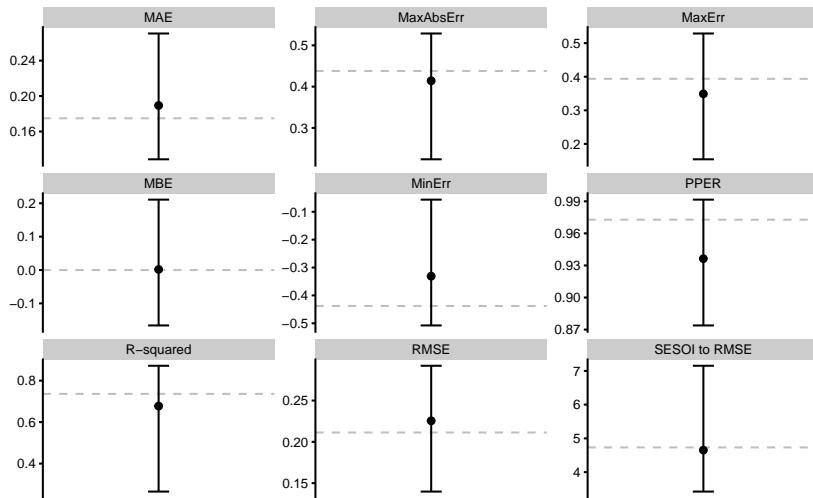


Figure 3.15: **Cross-validated model performance.** Dot and line bar indicate mean, min and max of the cross-validated performance. Dotted line indicate model performance on the training data set.

As can be seen from the Figure 3.15, cross-validated prediction performance metrics do not differ much¹¹ from the metrics estimated using the full training sample (calculated in [Describing relationship between two variables](#) section and in the Table 3.8, and indicated by the dotted horizontal line in the Figure 3.15). For some more complex models, these differences can be much larger and are clear indication of the model over-fitting.

Overall, predicting MAS from the YoYoIR1 score, *given* the data collected, SESOI of $\pm 0.5\text{km/h}$, and linear regression as a model, is practically excellent. Please note that prediction can be practically useful (given SESOI) (PPER; CV from 0.87 to 0.99) even when R-squared is relatively low (CV from 0.27 to 0.87). And the *vice versa* in some cases. That's the reason why we need to utilize magnitude-based estimators as a complement of contemporary estimators such as R-squared, RSE, and RMSE.

3.6.2 Predicting YoYoIR1 from MAS

As shown in [Describing relationship between two variables](#), predicting YoYoIR1 from MAS scores was not practically useful (or precise enough). But for the sake of completeness, let's perform cross-validated prediction (using 3 folds and 5 repeats). YoYoIR1 SESOI is taken to be $\pm 40\text{m}$.

¹¹As you will learn in [Statistical inference](#) section, we can perform statistical tests (in this case *t-test*) to check whether the average of the cross-validated performance metric differ from metric estimated on the full training sample.

Figure 3.16 depicts modified Bland-Altman plot for predictions using the full training data set. Visual inspection demonstrates that many points are outside of SESOI band, indicating poor practically significant (or useful) predictions.

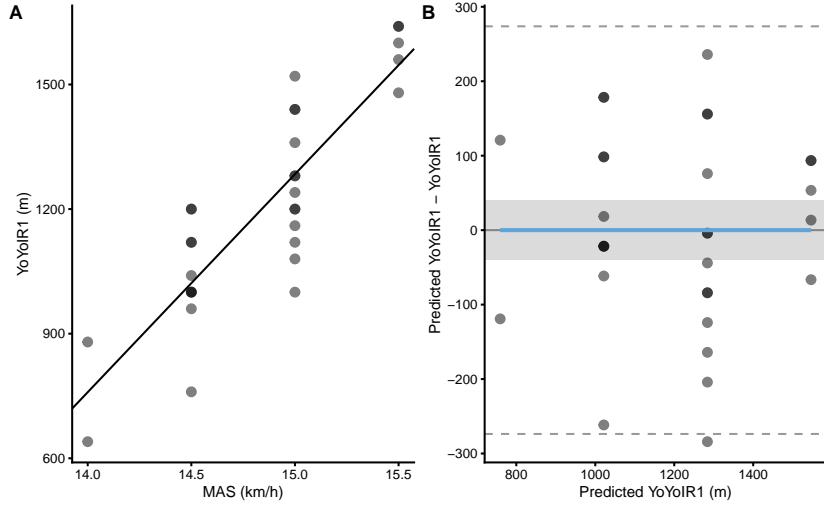


Figure 3.16: Scatter plot for simple linear regression between YoYoIR1 and MAS using the full training data sample. A. Scatter plot between YoYoIR1 and MAS scores. Black line indicates model prediction. B. Scatter plot between $y_{predicted}$ (fitted or predicted YoYoIR1) against model residual $y_{residual} = y_{predicted} - y_{observed}$, or Predicted YoYoIR1 - YoYoIR1. Dotted lines indicate Levels of Agreement (LOA; i.e. upper and lower threshold that contain 95% of residuals distribution) and grey band indicates SESOI. Blue line indicate linear regression fit of the residuals and is used to indicate issues with the model (residuals)

Predictive performance metrics can be found in the Table 3.10. As already expected, predicting YoYoIR1 from the MAS score, given the data collected, SESOI and linear regression model is not precise enough to be practically useful. Please note that the R-squared is very close to R-squared from the Table 3.10, but the PPER is much worse. Another reason to complement contemporary estimators with magnitude-based ones.

In the second part of this book, we will get back to this example and estimate predictive performance using different models besides linear regression (like baseline prediction and *regression trees*)

Table 3.11: Predictive performance summary

metric	training	training.pooled	testing.pooled	mean	SD	min	max
MBE	0.00	0.00	3.13	1.95	50.33	-96.20	82.14
MAE	104.69	103.04	112.98	112.59	24.99	66.61	154.76
RMSE	129.30	127.55	138.73	135.92	26.46	82.62	175.52
PPER	0.24	0.25	0.23	0.22	0.05	0.17	0.34
SESOI to RMSE	0.62	0.63	0.58	0.61	0.14	0.46	0.97
R-squared	0.74	0.74	0.70	0.70	0.14	0.48	0.88
MinErr	-235.93	-261.63	-253.04	-193.59	43.68	-253.04	-76.70
MaxErr	284.07	309.00	323.53	225.37	89.69	51.02	323.53
MaxAbsErr	284.07	309.00	323.53	260.89	44.88	158.47	323.53

Chapter 4

Causal inference

Does playing basketball makes one taller? This is a an example of a causal question. Wrestling with the concept of causality, as a philosophical construct is outside the scope of this book (and the author too), but I will define it using the *counterfactual theory* or *potential outcomes* perspective (Hernán, Hsu, and Healy 2019; Kleinberg 2015; Pearl and Mackenzie 2018; Angrist and Pischke 2015; Gelman 2011) that define causes in terms of how things would have been different had the cause not occurred, as well as from *causality-as-intervention* perspective (Gelman 2011), which necessitates clearly defined interventions (Hernán 2018, 2016; Hernán and Taubman 2008). In other words, would someone be shorter if basketball was never trained?

There are two broad classes of inferential questions that focus on *what if* and *why*: *forward causal inference* (“What might happen if we do X ?”) and *reverse causal inference* (“What causes Y ? Why?”) (Gelman 2011). Forward causation is more clearly defined problem, where the goal is to quantify the causal effect of treatment. Questions of forward causation are most directly studied using *randomization* (Gelman 2011) and are answered from the above mentioned causality-as-intervention and counterfactual perspectives. Reverse causation is more complex and it is more related to *explaining* the causal chains using the *system-variable* approach. Article by Gelman (2011) provides great overview of the most common causal perspectives, out of which I will mostly focus on forward causation.

4.1 Necessary versus sufficient causality

Furthermore, we also need to distinguish between four kinds of causation (Pearl and Mackenzie 2018; Kleinberg 2015): *necessary causation*, *sufficient causation* and neither or both. For example, if someone says that A causes B, then:

Table 4.1: Four kinds of causation

Cause	Necessary	Sufficient	Neither	Both
A happens	B might happen	B always happen	B might happen	B always happen
A doesn't happen	B never happens	B might happen	B might happen	B never happens

- If A is *necessary* for B, it means that if A never happened (counterfactual reasoning), then B will never happen. Or, in other words, B can never happen without A. But sufficient causality also means that A can happen without B happening.
- If A is *sufficient* for B, it means that if you have A, you will *always* have B. In other words, B always follows A. However, sometimes B can happen without A
- If A is *neither sufficient nor necessary* for B, then sometimes when A happens B will happen. B can also happen without A.
- If A is both necessary and sufficient for B, then B will always happen after A, and B will never happen without A.

Table 4.1 contains summary of the above necessary and sufficient causality. In all four types of causation, the concept of counterfactual reasoning is invoked.

Although the causal inference is a broad area of research, philosophical discussion and conflicts, there are a few key concepts that need to be introduced to get the big picture and understand the basics behind the aims of causal inference. Let's start with an example involving the aforementioned question whether playing basketball makes one taller.

4.2 Observational data

In order to answer this question, we have collected height data (expressed in cm) for the total of N=30 athletes, of which N=15 play basketball, and N=15 don't play basketball (Table 4.2). Playing basketball can be considered *intervention* or *treatment*, in which causal effect we are interested in. Basketball players are considered *intervention group* or *treatment group* and those without the treatment are considered *comparison group* or *control group*

Using descriptive estimators introduced in the [Description](#) section, one can quickly calculate the group `mean` and `SD` as well as their difference (Table 4.3). But does mean difference between basketball and control represent *average causal effect* (ACE)¹? No, unfortunately not!

¹Another term used is *average treatment effect* (ATE)

Table 4.2: Height in the treatment and control groups

Athlete	Treatment	Height (cm)
Athlete 27	Basketball	214
Athlete 01	Basketball	214
Athlete 25	Basketball	211
Athlete 19	Basketball	210
Athlete 03	Basketball	207
Athlete 21	Basketball	200
Athlete 23	Basketball	199
Athlete 15	Basketball	198
Athlete 17	Basketball	193
Athlete 07	Basketball	192
Athlete 29	Basketball	192
Athlete 13	Basketball	191
Athlete 05	Basketball	191
Athlete 11	Basketball	184
Athlete 09	Basketball	180
Athlete 02	Control	189
Athlete 28	Control	183
Athlete 06	Control	181
Athlete 14	Control	180
Athlete 04	Control	179
Athlete 12	Control	176
Athlete 18	Control	176
Athlete 08	Control	173
Athlete 26	Control	173
Athlete 24	Control	170
Athlete 30	Control	168
Athlete 20	Control	168
Athlete 16	Control	165
Athlete 10	Control	165
Athlete 22	Control	163

Table 4.3: Descriptive analysis of the groups

	Mean (cm)	SD (cm)
Basketball	198.59	10.86
Control	174.04	7.54
Difference	24.55	13.22

Table 4.4: Counterfactuals of potential outcomes that are unknown

Athlete	Treatment	Height_0 (cm)	Height_1 (cm)	Height (cm)	Causal Effect (cm)
Athlete 27	Basketball	???	214	214	???
Athlete 01	Basketball	???	214	214	???
Athlete 25	Basketball	???	211	211	???
Athlete 19	Basketball	???	210	210	???
Athlete 03	Basketball	???	207	207	???
Athlete 21	Basketball	???	200	200	???
Athlete 23	Basketball	???	199	199	???
Athlete 15	Basketball	???	198	198	???
Athlete 17	Basketball	???	193	193	???
Athlete 07	Basketball	???	192	192	???
Athlete 29	Basketball	???	192	192	???
Athlete 13	Basketball	???	191	191	???
Athlete 05	Basketball	???	191	191	???
Athlete 11	Basketball	???	184	184	???
Athlete 09	Basketball	???	180	180	???
Athlete 02	Control	189	???	189	???
Athlete 28	Control	183	???	183	???
Athlete 06	Control	181	???	181	???
Athlete 14	Control	180	???	180	???
Athlete 04	Control	179	???	179	???
Athlete 12	Control	176	???	176	???
Athlete 18	Control	176	???	176	???
Athlete 08	Control	173	???	173	???
Athlete 26	Control	173	???	173	???
Athlete 24	Control	170	???	170	???
Athlete 30	Control	168	???	168	???
Athlete 20	Control	168	???	168	???
Athlete 16	Control	165	???	165	???
Athlete 10	Control	165	???	165	???
Athlete 22	Control	163	???	163	???

4.3 Potential outcomes or counterfactuals

To explain why this is the case, we need to imagine *alternate counterfactual reality*. What is needed are two potential outcomes: $Height_0$, which represents height of the person if one doesn't train basketball, and $Height_1$ which represents height of the person if basketball is being played (Table 4.4). As can be guessed, the Basketball group has known $Height_1$, but unknown $Height_0$ and *vice versa* for the Control group.

Unfortunately, these potential outcomes are unknown, and thus individual causal effects are unknown as well. We just do not know what might have happened to individual outcomes in counterfactual world (i.e. alternate reality). A good control group serves as a *proxy* to reveal what might have happened *on average* to the treated group in the counterfactual world where they are not treated. Since the basketball data is simulated, the exact DGP is known (the *true* systematic

Table 4.5: Simulated causal effects and known counterfactuals

Athlete	Treatment	Height_0 (cm)	Height_1 (cm)	Height (cm)	Causal Effect (cm)
Athlete 27	Basketball	214	214	214	0.12
Athlete 01	Basketball	214	214	214	0.00
Athlete 25	Basketball	212	211	211	-1.10
Athlete 19	Basketball	210	210	210	0.90
Athlete 03	Basketball	208	207	207	-0.07
Athlete 21	Basketball	200	200	200	0.20
Athlete 23	Basketball	198	199	199	0.57
Athlete 15	Basketball	198	198	198	0.18
Athlete 17	Basketball	193	193	193	0.44
Athlete 07	Basketball	193	192	192	-0.09
Athlete 29	Basketball	193	192	192	-0.40
Athlete 13	Basketball	192	191	191	-0.36
Athlete 05	Basketball	191	191	191	-0.15
Athlete 11	Basketball	183	184	184	0.46
Athlete 09	Basketball	179	180	180	0.72
Athlete 02	Control	189	189	189	0.06
Athlete 28	Control	183	184	183	0.41
Athlete 06	Control	181	181	181	0.42
Athlete 14	Control	180	180	180	-0.66
Athlete 04	Control	179	179	179	0.10
Athlete 12	Control	176	176	176	-0.39
Athlete 18	Control	176	175	176	-0.31
Athlete 08	Control	173	173	173	-0.55
Athlete 26	Control	173	174	173	0.77
Athlete 24	Control	170	170	170	0.02
Athlete 30	Control	168	168	168	0.27
Athlete 20	Control	168	168	168	-0.03
Athlete 16	Control	165	165	165	-0.01
Athlete 10	Control	165	164	165	-0.81
Athlete 22	Control	163	163	163	0.00

or main causal effect of playing basketball on height is exactly zero), which again demonstrates the use of simulations as a great learning tool, in this case understanding the underlying causal mechanisms (Table 4.5). Individual causal effect in this case is the difference between two potential outcomes: $Height_1$ and $Height_0$.

From Table 4.5, we can state that the mean difference between the groups consists of two components: *average causal effect* and the *selection bias* (Angrist and Pischke 2015) (Equation (4.1)).

$$\begin{aligned}
 mean_{difference} &= Average\ causal\ effect + Selection\ bias \\
 Average\ causal\ effect &= \frac{1}{N_{Basketball}} \sum_{i=1}^n (Height_{1i} - Height_{0i}) \\
 Selection\ bias &= \frac{1}{N_{Basketball}} \sum_{i=1}^n Height_{0i} - \frac{1}{N_{Control}} \sum_{i=1}^n Height_{0i}
 \end{aligned} \tag{4.1}$$

The mean group difference we have observed (24.55cm) is due to average causal effect (0.1cm) and selection bias (24.46cm). In other words, observed mean group difference can be explained solely by selection bias. Since we know the DGP behind the basketball data, we know that there is no systematic causal effect of playing basketball on height.

On top of the selection bias involved in the example above, other *confounders* might be involved, such as age, sex, race, experience and others, some of which can be measured and some might be unknown. These are also referred to as the *third variable* which confounds the causal relationship between treatment and the outcome. In this example, all subjects from the Basketball group might be older males, whereas all the subjects from the Control group might be younger females, and this can explain the group differences, rather than causal effect of playing basketball.

4.4 *Ceteris paribus* and the biases

It is important to understand that, in order to have causal interpretation, comparisons need to be made under *ceteris paribus* conditions (Angrist and Pischke 2015), which is Latin for *other things equal*. In the basketball example above, we cannot make causal claim that playing basketball makes one taller, since comparison between the groups is not done in the *ceteris paribus* conditions due to the selection bias involved. We also know this since we know the DGP behind the observed data.

Causal inference thus aims to achieve *ceteris paribus* conditions needed to make causal interpretations by careful considerations of the known and unknown biases involved (Angrist and Pischke 2015; Hernán 2017, 2016; Hernán and Robins, n.d.; Hernán, Hsu, and Healy 2019; Lederer et al. 2019; Rohrer 2018; Shrier and Platt 2008).

According to Hernan *et al.* (Hernán 2017; Hernán and Robins, n.d.), there are three types of biases involved in causal inference: *confounding*, *selection bias* and *measurement bias*.

Confounding is the bias that arises when treatment and outcome share causes. This is because treatment was not randomly assigned (Hernán 2017; Hernán

and Robins, n.d.). For example, athletes that are naturally taller might be choosing to play basketball due to success and enjoyment over their shorter peers. On the other hand, it might be some hidden confounder that motivates *to-be-tall* athletes to choose basketball. Known and measured confounders from the observational studies can be taken into account to create *ceteris paribus* conditions when estimating causal effects (Angrist and Pischke 2015; Hernán 2017; Hernán and Robins, n.d.; Lederer et al. 2019; Rohrer 2018; Shrier and Platt 2008).

4.4.1 Randomization

The first line of defence against confounding and selection bias is to randomly assign athletes to treatment, otherwise known as *randomized trial* or *randomized experiment*. Random assignment makes comparison between groups *ceteris paribus* providing the sample is large enough to ensure that differences in the individual characteristics such as age, sex, experience and other potential confounders are *washed out* (Angrist and Pischke 2015). In other words, random assignment works not by eliminating individual differences but rather by ensuring that the mix of the individuals being compared is the same, including the ways we cannot easily measure or observe (Angrist and Pischke 2015).

In case the individuals from the basketball example were randomly assigned, given the known causal DGP, then the mean difference between the groups would be more indicative of the causal effect of playing basketball on height (Table 4.6).

If we calculate the mean differences in this randomly assigned basketball treatment (Table 4.7), we can quickly notice that random assignment washed out selection bias involved with the observational study, and that the mean difference is closer to the known systematic (or average or *expected*) causal effect. The difference between estimated systematic causal effect using mean group difference from the randomized trial and the true causal effect is due to the *sampling error* which will be explained in the [Statistical inference](#) section.

Apart from creating *ceteris paribus* conditions, randomization generates a good control group that serves as a *proxy* to reveal what might have happened to the treated group in the counterfactual world where they are not treated, since $Height_0$ is not known for the basketball group. Creating those conditions with randomized trial demands careful considerations and *balance checking* since biases can *crawl* inside the causal interpretation. The logic of randomized trial is simple, yet the logistics can be quite complex. For example, a sample of sufficient size might not be practically feasible, and imbalances in the known confounders can be still found in the groups, thus demanding further control and adjustment in the analysis (e.g. using ANCOVA instead of ANOVA, adjusting for confounders in the linear regression by introducing them as interactions) in order to create *ceteris paribus* conditions needed to evaluate causal claims.

Table 4.6: Randomized participants

Athlete	Treatment	Height (cm)
Athlete 01	Basketball	214
Athlete 25	Basketball	211
Athlete 19	Basketball	210
Athlete 03	Basketball	207
Athlete 23	Basketball	199
Athlete 15	Basketball	198
Athlete 13	Basketball	191
Athlete 02	Basketball	189
Athlete 28	Basketball	184
Athlete 14	Basketball	180
Athlete 04	Basketball	179
Athlete 12	Basketball	176
Athlete 26	Basketball	174
Athlete 24	Basketball	170
Athlete 16	Basketball	165
Athlete 27	Control	214
Athlete 21	Control	200
Athlete 29	Control	193
Athlete 07	Control	193
Athlete 17	Control	193
Athlete 05	Control	191
Athlete 11	Control	183
Athlete 06	Control	181
Athlete 09	Control	179
Athlete 18	Control	176
Athlete 08	Control	173
Athlete 30	Control	168
Athlete 20	Control	168
Athlete 10	Control	165
Athlete 22	Control	163

Table 4.7: Descriptive summary of randomized participants

	Mean (cm)	SD (cm)
Basketball	189.91	16.15
Control	182.65	14.44
Difference	7.25	21.66

Belief effect can sneak in, for example, if the treatment group *knows* they are being treated, or if researchers motivate treatment groups harder, since they expect and hope for better outcomes. For this reason, *blinding* both the subjects and researchers can be considered, as well as providing *placebo* treatment to the Control group. In sport science research blinding and providing placebo can be problematic. For example, if our intervention is a novel training method or a technology, both researchers and subjects will expect better outcomes which can bias causal interpretations.

4.5 Subject matter knowledge

One of the main problems with randomized trials is that it cannot be done in most real life settings, either due to the ethical or practical reasons. For example, if studying effects of smoking on baby mortality and birth defects, which parent would accept being in the treatment group. Or if studying effects of resistance training on injury risk in football players, which professional organization would allow random assignment to the treatment that is lesser than the known best practices and can predispose athletes to the injuries or sub-par preparation?

For this reason, reliance on observation studies is the best we can do. However, in order to create *ceteris paribus* conditions necessary to minimize bias in the causal interpretations, expert subject-matter knowledge is needed, not only to describe the causal structure of the system under study, but also to specify the causal questions and identify relevant data sources (Hernán, Hsu, and Healy 2019). Imagine asking the following causal question: “Does training load lead to overuse injuries in professional sports”. It takes expert subject matter knowledge to specify the treatment construct (i.e. “training load”), to figure out how should be measured, as well as to quantify the measurement error which can induce *measurement bias*, to state over which time period the treatment is done, as well as to specify the outcome construct (i.e. “overuse-injuries”), and to define the variables and constructs that confound and define the causal network underlying such a question. This subject matter is fallible of course, and the constructs, variables and the causal network can be represented with pluralistic models that represents “Small World” maps of the complex “Large World”, in which we are hoping to deploy the findings (please refer to the [Introduction](#) for more information about this concept). Drawing assumptions that underly causal structure using *direct acyclical graphs* (DAGs) (Hernán 2017; Hernán and Robins, n.d.; Pearl and Mackenzie 2018; Rohrer 2018; Saddiki and Balzer 2018; Shrier and Platt 2008; Textor et al. 2017) represents a step forward in acknowledging the issues above, by providing transparency of the assumptions involved and bridging the subjective - objective dichotomy.

4.6 Example of randomized control trial

Let's consider the following example. We are interested in estimating causal effect of the plyometric training on the vertical jump height. To estimate causal effect, *randomized control trial* (RCT) is utilized. RCT utilizes two groups: Treatment ($N=15$) and Control ($N=15$), measured two times: Pre-test and Post-test. Treatment group received plyometric training over the course of three months, while Control group continued with *normal* training. The results of RCT study can be found in the Table 4.8. To estimate practical significance of the treatment effect, SESOI of $\pm 2.5\text{cm}$ is selected to indicate minimal change of the practical value. It is important to have “well defined interventions” (Hernán 2018, 2016; Hernán and Taubman 2008), thus the question that should be answered is as follows: “Does plyometric training added to normal training improves vertical jump height over period of three months?”

Descriptive summary statistics for Treatment and Control groups are enlisted in the Table 4.9, and visually depicted in the Figure 4.1.

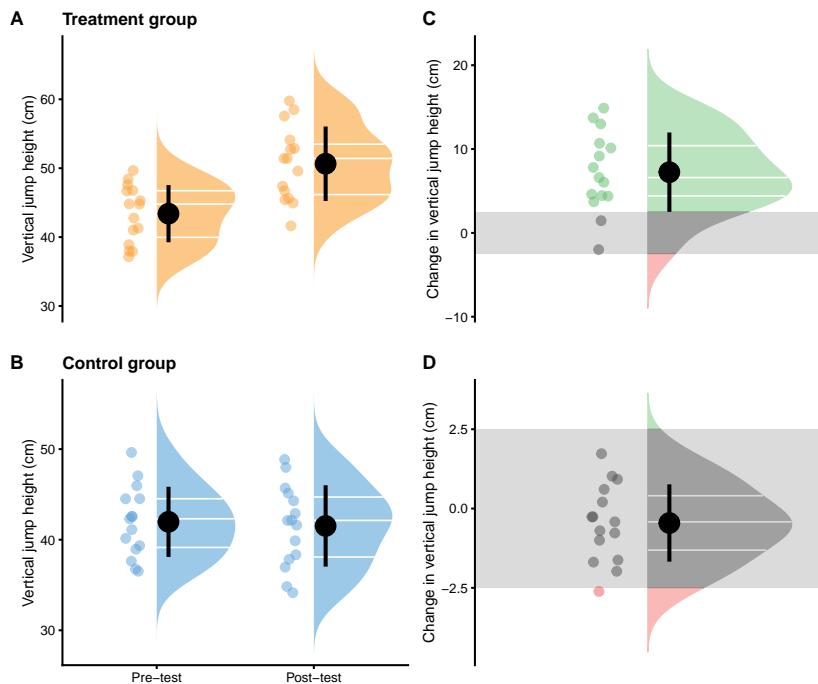


Figure 4.1: **Visual analysis of RCT using Treatment and Control groups.** **A and B.** Raincloud plot of the Pre-test and Post-test scores for Treatment and Control groups. Blue color indicates Control group and orange color indicates Treatment group. **C and D.** Raincloud plot of the change scores for the Treatment and Control groups. SESOI is indicated with a grey band

Table 4.8: Randomized control trial data

Athlete	Group	Pre-test (cm)	Post-test (cm)	Change (cm)
Athlete 01	Treatment	37.98	52.86	14.93
Athlete 27	Treatment	44.79	58.50	13.43
Athlete 19	Treatment	46.77	59.76	12.99
Athlete 25	Treatment	38.90	49.58	10.81
Athlete 03	Treatment	41.29	51.41	10.34
Athlete 23	Treatment	48.41	57.57	8.58
Athlete 17	Treatment	44.81	51.41	7.85
Athlete 21	Treatment	37.14	44.95	7.37
Athlete 15	Treatment	46.69	52.73	6.14
Athlete 29	Treatment	42.77	47.38	5.02
Athlete 13	Treatment	49.66	54.11	4.46
Athlete 05	Treatment	37.92	41.63	3.78
Athlete 07	Treatment	41.03	45.41	3.42
Athlete 11	Treatment	45.27	46.72	1.82
Athlete 12	Control	42.56	44.29	1.01
Athlete 28	Control	47.06	47.98	0.55
Athlete 04	Control	44.53	45.13	0.12
Athlete 02	Control	49.63	48.86	-0.01
Athlete 08	Control	41.11	42.13	-0.45
Athlete 26	Control	42.31	41.61	-0.51
Athlete 06	Control	45.96	45.70	-0.52
Athlete 14	Control	44.51	42.89	-0.63
Athlete 18	Control	42.57	42.15	-0.74
Athlete 16	Control	37.63	37.83	-0.75
Athlete 22	Control	36.52	34.83	-0.97
Athlete 24	Control	40.15	39.88	-1.03
Athlete 30	Control	39.34	38.34	-1.21
Athlete 09	Treatment	47.61	45.62	-1.57
Athlete 20	Control	38.94	36.97	-1.72
Athlete 10	Control	36.77	34.15	-2.26

Table 4.9: RCT summary using mean \pm SD

Group	Pre-test (cm)	Post-test (cm)	Change (cm)
Treatment	43.4 ± 4.15	50.64 ± 5.4	7.29 ± 4.65
Control	41.97 ± 3.87	41.52 ± 4.49	-0.61 ± 0.83

Table 4.10: Descriptive analysis of the change scores for Treatment and Control groups independently

Estimator	Control	Treatment
Mean change (cm)	-0.61	7.29
SDchange (cm)	0.83	4.65
SDpre-test pooled (cm)	4.01	4.01
Cohen's d	-0.15	1.82
SESOI lower (cm)	-2.50	-2.50
SESOI upper (cm)	2.50	2.50
Change to SESOI	-0.12	1.46
SDchange to SESOI	0.17	0.93
pLower	0.06	0.03
pEquivalent	0.93	0.14
pHigher	0.01	0.83

Further analysis might involve separate dependent groups analysis for both Treatment and Control (Table 4.10), or in other words, the analysis of the change scores. To estimate Cohen's d, pooled SD of the Pre-test scores in both Treatment and Control is utilized. (see Equation (2.11)).

Figure 4.2 depicts same information as Figure 4.1 but organized differently and conveying different comparison.

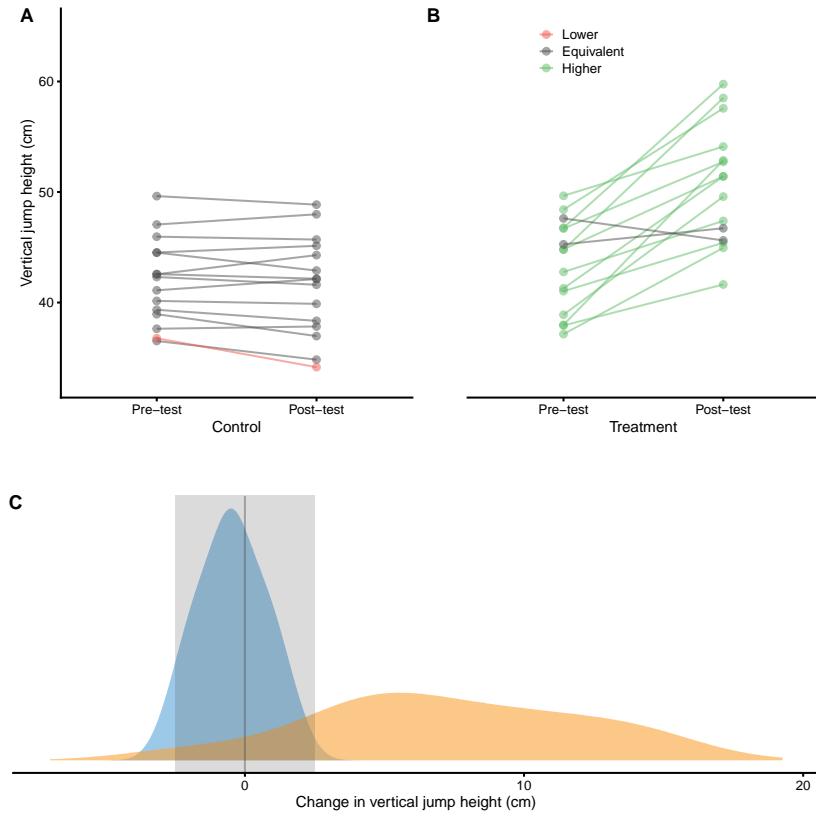


Figure 4.2: Visual analysis of RCT using Treatment and Control groups. **A and B.** Scatter plot of Pre-test and Post-test scores for Treatment and Control groups. Green line indicates change higher than SESOI upper, grey line indicates change within SESOI band, and red line indicates negative change lower than SESOI lower. **C.** Distribution of the change scores for Treatment (orange) and Control (blue) groups. Grey rectangle indicates SESOI band.

But we are not that interested in independent analysis of Treatment and Control groups, but rather on their differences and understanding of the causal effect of the treatment (i.e. understanding and estimating parameters of the underlying DGP). As stated, treatment effect consists of two components: systematic component or main effect (i.e. expected or average causal effect), and stochastic component or random effect (i.e. that varies between individuals) (see Figure 4.3). As already explained, Control group serves as a proxy to what might have happened to the Treatment group in the counterfactual world, and thus allows for casual interpretation of the treatment effect. There are two effects at play with this RCT design: *treatment effect* and *non-treatment effect*. The

latter captures all effects not directly controlled by a treatment, but assumes it affects both groups equally (Figure 4.3). For example, if we are treating kids for longer period of time, non-treatment effect might be related to the growth and associated effects. Another non-treatment effect is *measurement error* (discussed in more details in [Measurement Error](#) section).

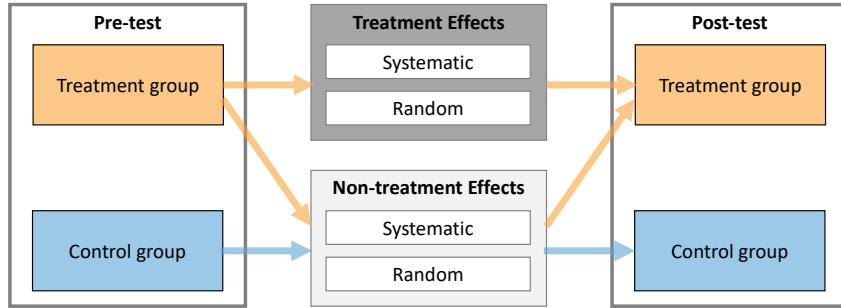


Figure 4.3: Treatment and Non-treatment effects of intervention. Both treatment and non-treatment effects consists of two components: systematic and random. Treatment group experiences both treatment and non-treatment effects, while Control group experiences only non-treatment effects.

The following equation captures the essence of estimating Treatment effects from Pre-test and Post-test scores in the Treatment and Control groups (Equation (4.2)):

$$\begin{aligned} Treatment_{post} &= Treatment_{pre} + Treatment\ Effect + NonTreatment\ Effect \\ Control_{post} &= Control_{pre} + NonTreatment\ Effect \end{aligned}$$

$$NonTreatment\ Effect = Control_{post} - Control_{pre}$$

$$Treatment\ Effect = Treatment_{post} - Treatment_{pre} - NonTreatment\ Effect$$

$$Treatment\ Effect = (Treatment_{post} - Treatment_{pre}) - (Control_{post} - Control_{pre})$$

$$Treatment\ Effect = Treatment_{change} - Control_{change}$$

(4.2)

From the Equation (4.2), the differences between the changes in Treatment and Control groups can be interpreted as the estimate of the causal effect of the treatment. More precisely, average causal effect or expected causal effect represent systematic treatment effect. This is estimated using difference between **mean** Treatment change and **mean** Control change.

Table 4.11: Descriptive statistics of the change score differences

Mean difference (cm)	Cohen's d	Difference to SESOI	pLower diff	pEquivalent diff	pHigher diff
7.9	9.56	1.58	-0.03	-0.79	0.82

Table 4.11 contains descriptive statistics of the change score differences. Panel C in the Figure 4.2 depicts distribution of the change scores and reflect the calculus in the Table 4.11 graphically.

Cohen's d in the Table 4.11 is calculated by using the Equation (4.3) and it estimates standardized difference between change scores in Treatment and the Control groups.

$$\text{Cohen's } d = \frac{\text{mean}_{\text{treatment group change}} - \text{mean}_{\text{control group change}}}{SD_{\text{control group change}}} \quad (4.3)$$

Besides estimating systematic component of the treatment (i.e. the difference between the mean change in Treatment and Control groups), we might be interested in estimating random component and proportions of lower, equivalent and higher effects compared to SESOI (pLower, pEquivalent, and pHiger). Unfortunately, differences in pLower, pEquivalent, and pHiger from Table 4.11 don't answer this question, but rather the expected difference in proportions compared to Control (e.g. the expected improvement of 0.82 in observing proportion of higher change outcomes compared to Control).

Since the changes in Treatment group are due both to the treatment and non-treatment effects (equation 29), the average treatment effect (systematic component) represents the difference between the mean changes in Treatment and Control groups (Table 4.11). In the same manner, the variance of the change scores in the Treatment group are due to the random component of the treatment and non-treatment effects. Assuming normal (Gaussian) distribution of the random components, the SD of the treatment effects (SD_{TE})² is estimated using the following Equation (4.4).

²Also referred to as SD_{IR} or standard deviation of the intervention responses (Will G. Hopkins 2015; Swinton et al. 2018). SD_{IR} or SD_{TE} represent estimate of treatment effect heterogeneity, also referred to as variable treatment effect (VTE)

$$\begin{aligned}
\epsilon_{treatment \ group \ change} &= \epsilon_{treatment \ effect} + \epsilon_{nontreatment \ effect} \\
\epsilon_{control \ group \ change} &= \epsilon_{nontreatment \ effect} \\
\epsilon_{treatment \ effect} &= \epsilon_{treatment \ group \ change} - \epsilon_{control \ group \ change} \\
\epsilon_{treatment \ effect} &\sim \mathcal{N}(0, SD_{TE}) \\
\epsilon_{nontreatment \ effect} &\sim \mathcal{N}(0, SD_{NTE}) \\
\epsilon_{treatment \ group \ change} &\sim \mathcal{N}(0, SD_{treatment \ group \ change}) \\
\epsilon_{control \ group \ change} &\sim \mathcal{N}(0, SD_{control \ group \ change})
\end{aligned}$$

$$SD_{TE} = \sqrt{SD_{treatment \ group \ change}^2 - SD_{control \ group \ change}^2} \quad (4.4)$$

This neat mathematical solution is due to assumption of Gaussian error, assumption that random treatment and non-treatment effects are equal across subjects (see [Ergodicity](#) section for more details about this assumption), and the use of squared errors. This is one beneficial property of using squared errors that I alluded to in the section [Cross-Validation](#) section.

Thus, the estimated parameters of the causal treatment effects in the underlying DGP are summarized with the following Equation (4.5). This treatment effect is graphically depicted in the Figure 4.4.

$$Treatment \ effect \sim \mathcal{N}(Mean_{TE}, SD_{TE})$$

$$Mean_{TE} = Mean_{treatment \ group \ change} - Mean_{control \ group \ change}$$

$$SD_{TE} = \sqrt{SD_{treatment \ group \ change}^2 - SD_{control \ group \ change}^2} \quad (4.5)$$

Table 4.12: Estimates of the causal treatment effects

Average causal effect (cm)	Random effect (cm)	SESOI (cm)	Average causal effect to SESOI	SESOI to random effect	pLower	pEquivalent	pHigher
7.9	4.57	± 2.5	1.58	1.09	0.01	0.11	0.88

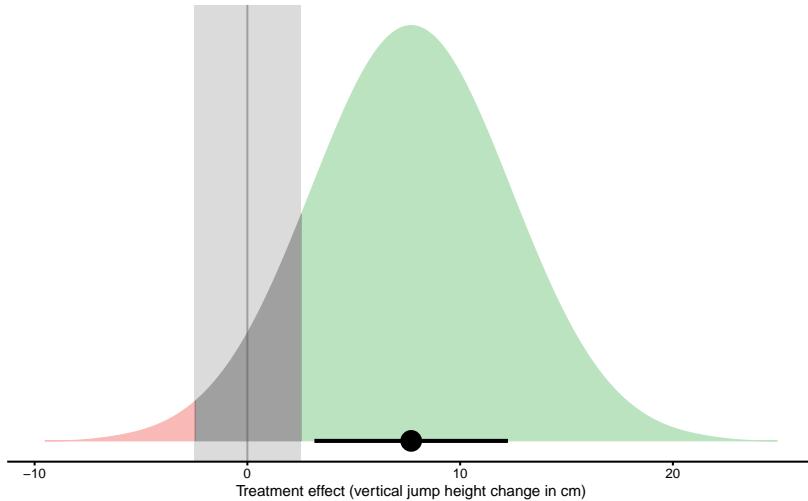


Figure 4.4: **Graphical representation of the causal Treatment effect.** Green area indicates proportion of higher than SESOI treatment effects, red indicates proportion of negative and lower than SESOI treatment effects, and grey indicates treatment effects that are within SESOI. Mean of treatment effect distribution represents average (or expected) causal effect or systematic treatment effect. SD of treatment effect distribution represents random systematic effect or SD_{TE}

Using SESOI, one can also estimate the proportion of lower, equivalent and higher changes (responses) caused by treatment. The estimates of the causal treatment effects, with accompanying proportions of responses are enlisted in the Table 4.12.

Therefore, we can conclude that plyometric training over three months period, on top of the normal training, cause improvements in vertical jump height (in the sample collected; *generalizations* beyond sample are discussed in the [Statistical inference](#) section). The expected improvement (i.e. average causal effect or systematic effect) is equal to 7.9cm, with 1, 11, and 88% of athletes having lower, trivial and higher improvements.

4.7 Prediction as a complement to causal inference

In the previous section, RCT is analyzed using *analysis of changes*. In this section, I will utilize linear regression model to analyze RCT data. There are multiple ways this could be done (J et al. 2018) and deeper analysis is beyond the scope of this book (see also Frank Harrell³ post on the use of change scores). The aim of this section is to provide an introduction to *model-based* and *prediction-based* RCT analysis, as well as to demonstrate potential uses of PDP+ICE plots as tools for counterfactual analysis.

Analysis in the previous section can be represented using simple linear regression model (Equation (4.6)).

$$\widehat{Change} = \hat{\beta}_0 + \hat{\beta}_1 Group \quad (4.6)$$

According to Frank Harrell⁴, the use of change scores in the RCT analysis is problematic. Although this model definition (Equation (4.6)) will give us exactly the same results as obtained in the previous section, the use of change scores should be avoided. Thus, look at this example as *training vehicle*. After this initial discussion, valid model representation will be used.

Since Group column is a string, how is Group column represented in the model? Group column needs to be *dummy-coded*, using 0 for Control and 1 for Treatment (see Table 4.13).

Estimated parameters for this linear model are enlisted in the Table 4.14

Intercept in the Table 4.14 represents the `mean` Change in the Control group, while $\hat{\beta}_1$ (or slope, or `GroupTreatment` parameter) represents estimated average treatment effect (ATE) or average causal effect, since it represents the difference in the Change means between groups. For a reference please refer to Tables 4.10 and 4.12.

Figure 4.5 depicts this model graphically.

³<https://www.fharrell.com/post/errmed/#change>

⁴<https://www.fharrell.com/post/errmed/#change>

Table 4.13: **Dummy coding of the Group column to be used in linear regression model**

Athlete	groupTreatment	Pre-test (cm)	Post-test (cm)	Change (cm)
Athlete 12	0	42.56	44.29	1.01
Athlete 28	0	47.06	47.98	0.55
Athlete 04	0	44.53	45.13	0.12
Athlete 02	0	49.63	48.86	-0.01
Athlete 08	0	41.11	42.13	-0.45
Athlete 26	0	42.31	41.61	-0.51
Athlete 06	0	45.96	45.70	-0.52
Athlete 14	0	44.51	42.89	-0.63
Athlete 18	0	42.57	42.15	-0.74
Athlete 16	0	37.63	37.83	-0.75
Athlete 22	0	36.52	34.83	-0.97
Athlete 24	0	40.15	39.88	-0.27
Athlete 30	0	39.34	38.34	-1.21
Athlete 20	0	38.94	36.97	-1.72
Athlete 10	0	36.77	34.15	-2.26
Athlete 01	1	37.98	52.86	14.93
Athlete 27	1	44.79	58.50	13.43
Athlete 19	1	46.77	59.76	12.99
Athlete 25	1	38.90	49.58	10.81
Athlete 03	1	41.29	51.41	10.34
Athlete 23	1	48.41	57.57	8.58
Athlete 17	1	44.81	51.41	7.85
Athlete 21	1	37.14	44.95	7.37
Athlete 15	1	46.69	52.73	6.14
Athlete 29	1	42.77	47.38	5.02
Athlete 13	1	49.66	54.11	4.46
Athlete 05	1	37.92	41.63	3.78
Athlete 07	1	41.03	45.41	3.42
Athlete 11	1	45.27	46.72	1.82
Athlete 09	1	47.61	45.62	-1.57

Table 4.14: **Estimated linear regression parameters for the simple RCT model**

Intercept	groupTreatment
-0.61	7.9

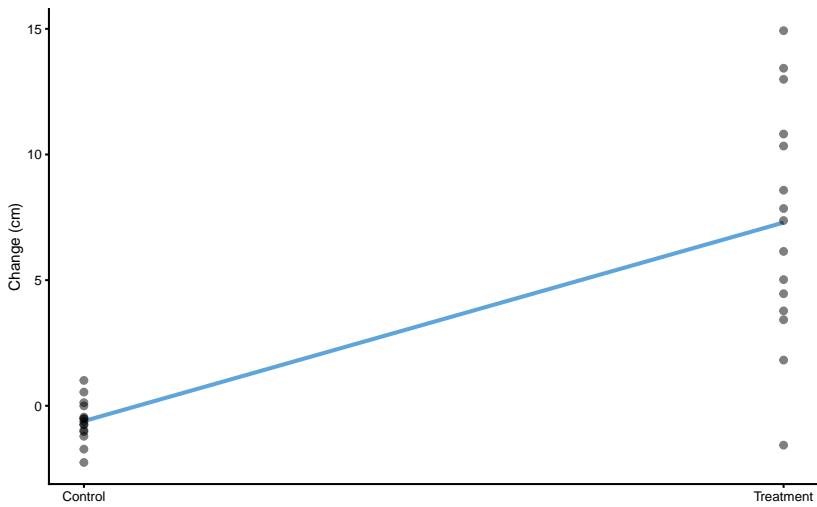


Figure 4.5: Graphical representation of the simple linear regression model for the vertical jump RCT data

Model residuals are depicted on Figure 4.6. Please note the *clusters* of the data-points which indicate groups (they are color-coded).

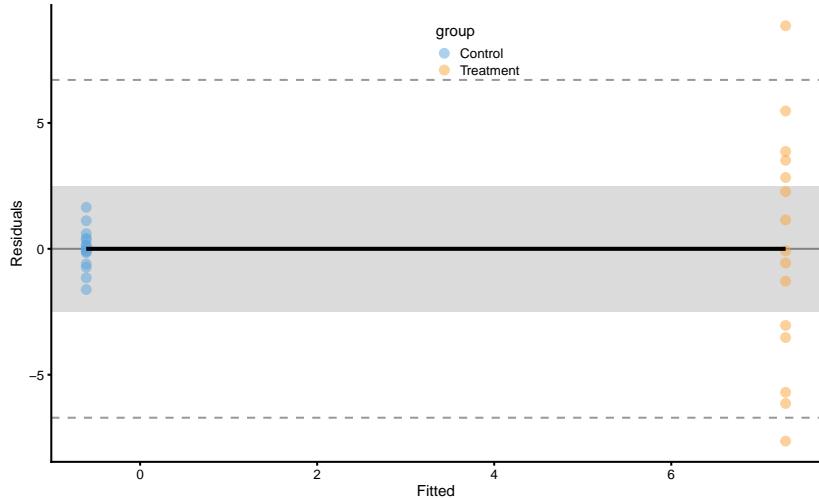


Figure 4.6: Model residuals using simple linear regression RCT model. Grey band represents SESOI of $\pm 2.5\text{cm}$. Residuals are color coded; blue are Control group and orange are Treatment group.

SD of the residuals for the Control group is equal to 0.83cm and for Treatment

group is equal to 4.65cm. Please compare these estimates and estimated parameters from the Table 4.14 with Table 4.10 and Table 4.12. These estimates are identical since the model utilized (Equation (4.6)) is mathematically equivalent to the analysis done in the [Example of randomized control trial](#) section.

As alluded in the introduction of this section, RCT analysis using change scores should be avoided. Valid way to analyze the RCT in this case is to use Post-test as the outcome, and Pre-test and Group as predictors. This can be easily understood graphically (Figure 4.7). On Figure 4.7 each group (i.e. Control and Treatment) is modeled separately.

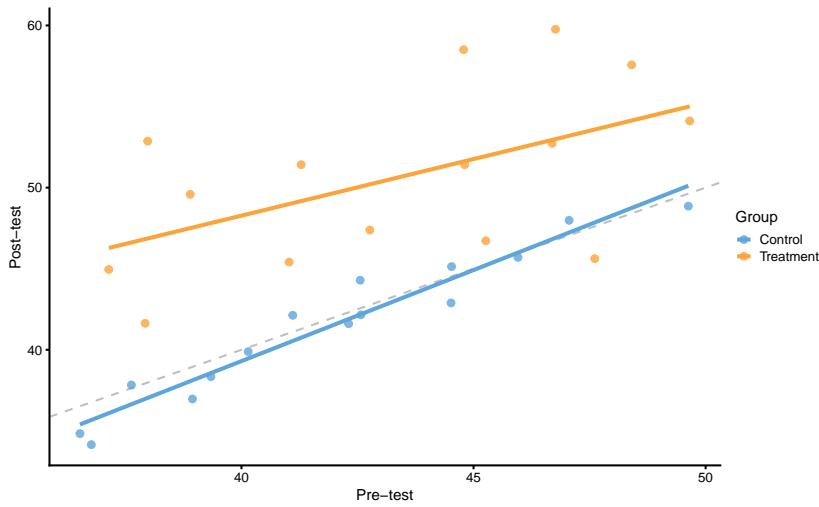


Figure 4.7: **Graphical representation of the valid way to analyze RCT data.** Dashed line represent *identity line*, where Post-test is equal to Pre-test (i.e., *the no effect line*). The effect of treatment represents vertical distance between the Control and Treatment lines. This is easily grasped since the lines are almost perfectly parallel. If the lines are not parallel, that would imply there is *interaction* between Group and Pre-test (i.e. individuals with higher Pre-test scores shows higher or lower change).

Figure 4.7 also represents ANCOVA (analysis of co-variance) design. Equation (4.7) represent model definition where effects of Group (i.e., Treatment) are estimated by controlling the effects of the Pre-test.

$$\widehat{Post} = \hat{\beta}_0 + \hat{\beta}_1 Group + \hat{\beta}_2 Pre \quad (4.7)$$

Estimated parameters for this linear model are enlisted in the Table 4.15. Please note the similarity with the Table 4.14.

Table 4.15: Estimated linear regression parameters for the ANCOVA RCT model (see Equation (4.7))

Intercept	groupTreatment	Pre-test
3.92	7.85	0.9

Model residuals are depicted on Figure 4.8. Please note the *clusters* of the data-points which indicate groups.

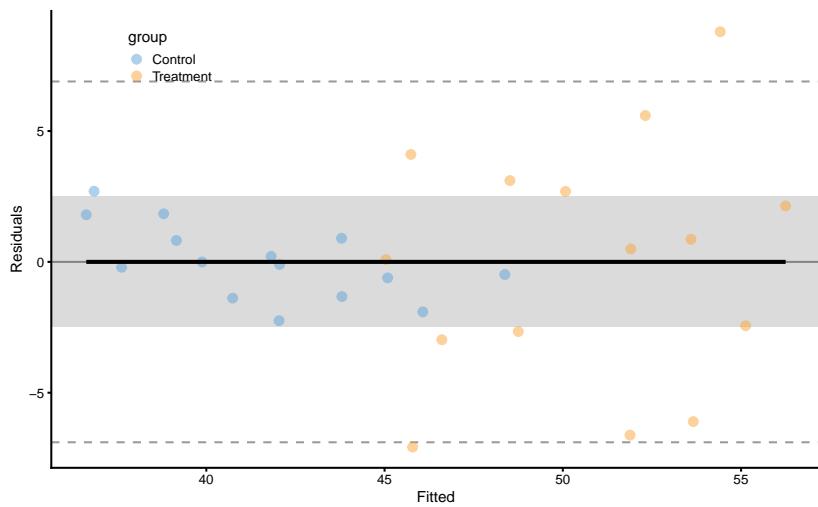


Figure 4.8: Model residuals using ANCOVA RCT model. Grey band represents SESOI of $\pm 2.5\text{cm}$. Residuals are color coded; blue are Control group and orange are Treatment group.

SD of the residuals for the Control group is equal to 1.43cm and for Treatment group is equal to 4.64cm. Please note the similarities with simple RCT model (i.e. using Change score as outcome and Group as predictor).

As explained in the [Prediction](#) section, this model can be cross-validated. Predictive performance metrics using 10 repeats of 3 folds cross-validation are enlisted in the Table 4.16. Since our RCT data has two groups (i.e. Control and Treatment), cross-validation needs to be *stratified*. This makes sure that each group has their own cross-validation folds, and that testing data size for each group is proportional to the group size. This avoids scenarios where most training or testing data comes from a single group (which is more probable if one group is larger).

From the Table 4.16 we can conclude, that although we have *explained* the

Table 4.16: Cross-validated predictive performance metrics for the AN-COVA RCT model

metric	training	training.pooled	testing.pooled	mean	SD	min	max
MBE	0.00	0.00	-0.04	-0.04	1.19	-2.00	2.04
MAE	2.41	2.37	2.69	2.69	0.55	1.68	4.39
RMSE	3.31	3.24	3.68	3.62	0.68	2.28	5.28
PPER	0.54	0.56	0.50	0.48	0.08	0.34	0.68
SESOI to RMSE	1.51	1.54	1.36	1.43	0.27	0.95	2.19
R-squared	0.75	0.76	0.69	0.70	0.14	0.25	0.91
MinErr	-7.08	-7.80	-8.83	-6.14	2.23	-8.83	-1.65
MaxErr	8.80	9.93	10.75	5.95	2.87	1.35	10.75
MaxAbsErr	8.80	9.93	10.75	7.91	1.59	3.78	10.75

Table 4.17: Cross-validated predictive performance metrics for the simple RCT model

metric	training	training.pooled	testing.pooled	mean	SD	min	max
MBE	0.00	0.00	0.00	0.00	1.05	-2.22	1.98
MAE	2.16	2.14	2.30	2.30	0.48	1.33	3.44
RMSE	3.22	3.19	3.41	3.35	0.66	2.02	4.53
PPER	0.55	0.57	0.53	0.51	0.09	0.39	0.73
SESOI to RMSE	1.55	1.57	1.47	1.56	0.35	1.10	2.48
R-squared	0.60	0.61	0.55	0.57	0.19	-0.14	0.82
MinErr	-7.64	-8.73	-9.10	-5.97	2.10	-9.10	-1.40
MaxErr	8.86	9.84	10.33	5.67	3.05	0.59	10.33
MaxAbsErr	8.86	9.84	10.33	7.69	1.62	3.78	10.33

causal (or treatment) effects, predicting individual Post-test is not practically meaningful since the prediction error is too large (compared to SESOI). The take-home message is that high *explanatory power* of the model doesn't automatically yield high predictive power (Shmueli 2010). The selection of statistical analysis is thus related to the question asked. In my opinion, it would be insightful to complement causal estimates with prediction estimates. With the example above, we can predict the direction of the effect (using expected systematic change of 7.9cm and proportions of 1, 11, and 88% for lower, trivial and higher change magnitudes), but we are unable to predict individual Post-test (or changes scores) within acceptable practical precision (using SESOI as an anchor). In other words, we know that the effect will be 88% beneficial (i.e. higher than SESOI), but we are not able to predict individual responses.

For the sake of completeness, Table 4.17 contains performance metrics for the simple RCT model.

4.7.1 Analysis of the individual residuals: responders vs non-responders

One particular use of the predictive analysis is in the identification of responders and non-responders to the treatment (Hecksteden et al. 2015, 2018; Will G. Hopkins 2015; Swinton et al. 2018). Common approach used in sport science (Will G Hopkins 2004a), that I will name *observed outcome approach* (further discussed in Measurement error chapter and second part of this book), uses known SESOI and *measurement error* to estimate probability of lower, equivalent, and higher changes. In RCT, random non-treatment effect can be assumed to be due to measurement error. Figure 4.9 depicts individual *adjusted change* (by deducting mean Control group change from observed change) with error bars representing *smallest detectable change* (SDC). SDC is calculated by multiplying Control group change SD by 1.96 (to get upper and lower change levels containing 95% of change distribution). This thus represent our *uncertainty* in true treatment effect (using Control group as source of information about random effects).

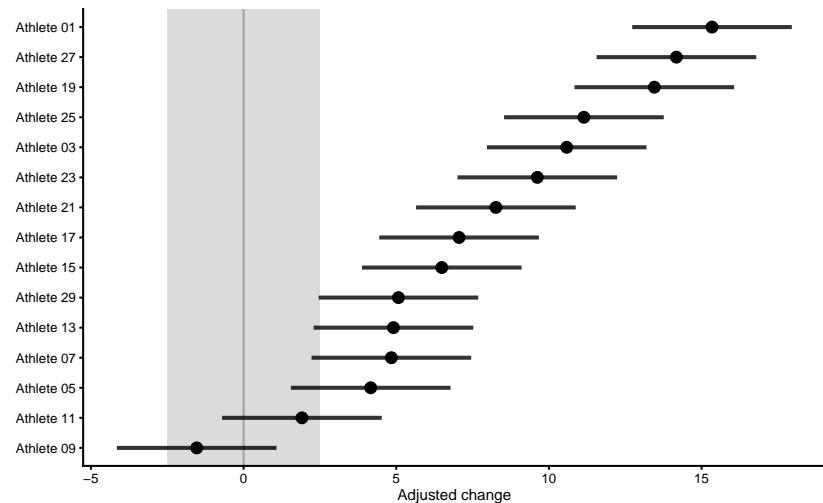


Figure 4.9: Responses analysis for the Treatment group. Change scores are adjusted by deducting Control group mean change. Error bars represent *smallest detectable change* (SDC) that is calculated using SD of the Control group change scores multiplied by 1.96 (to get 95% levels containing 95% of the change distribution).

Using this approach, we can classify athletes with high probability of higher change score as responders, those with high probability of equivalent change score as non-responders, and finally those with high probability of lower change score as negative-responders. This approach is useful in figuring out who responded positively or negatively to a particular treatment, but it doesn't take into account

information that might help explain the response (for example someone missing treatment session or having lower or higher treatment dose; see [Direct and indirect effect, covariates and then some](#) section). The topic is further discussed in [Measurement error](#) chapter and second part of this book.

Another approach, that I have termed *residuals approach* or *model-based approach* can be used to help identifying *outliers* to intervention. To explain this approach, let's plot athletes' residuals ($\hat{y}_i - y_i$) against observed Post-test (y_i) (Figure 4.10) using ANCOVA RCT model.

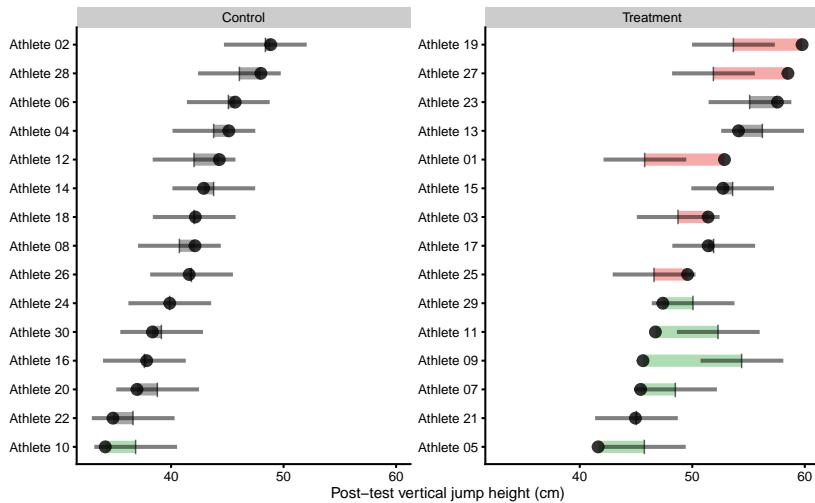


Figure 4.10: Observed Post-test in vertical jump for each athlete in the study. Black dot indicate observed Post-test value; vertical line indicate model prediction; colored bar indicated the residual magnitude compared to defined SESOI ($\pm 2.5\text{cm}$ in this example): grey for equivalent magnitude, green for lower magnitude, and red for higher magnitude; horizontal error bar represents cross-validated RMSE (see Table 4.16, RMSE metric, column *testing.pooled*) and is used to indicate model predictive performance and uncertainty around model prediction graphically

If we visualize simple model of RCT, using Change score as outcome and Group as predictor (see Figure 4.5), the predictions for athletes in each group are identical (i.e. the average change). This is depicted in Figure (Figure 4.11).

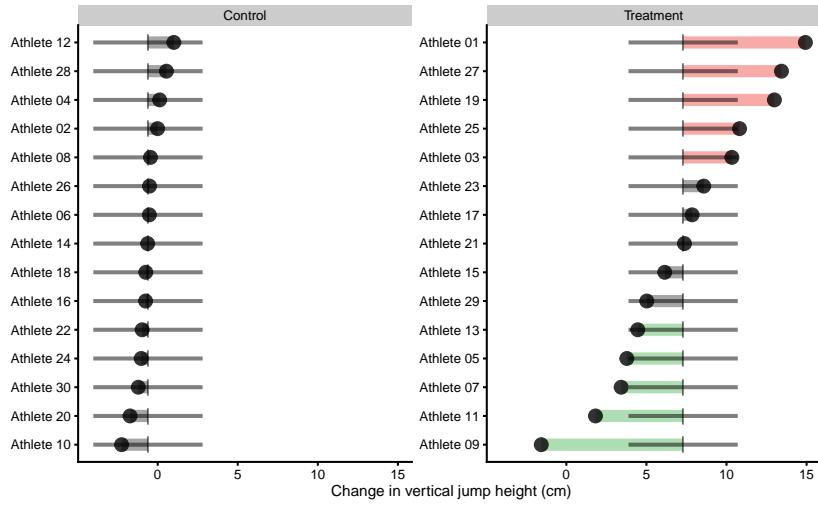


Figure 4.11: Observed Change in vertical jump for each athlete in the study. Black dot indicate observed Change value; vertical line indicate model prediction; colored bar indicated the residual magnitude compared to defined SESOI ($\pm 2.5\text{cm}$ in this example): grey for equivalent magnitude, green for lower magnitude, and red for higher magnitude; horizontal error bar represents cross-validated RMSE (see Table 4.17, RMSE metric, column *testing.pooled*) and is used to indicate model predictive performance and uncertainty around model prediction graphically

More complex models (e.g. ANCOVA RCT model in Figure 4.10), like the one utilized in [Direct and indirect effect, covariates and then some](#) section will have different predictions for each athlete.

Residuals approach uses observed scores and model predictions to indicate individuals who differ more or less than predicted by the model. If this difference between observed and predicted scores (or residual) is bigger than SESOI, this individual is *flagged*. But before jumping to conclusions, I need to remind you that the predictive performance of this simple model is pretty bad (see Table 4.17). Thus, this type of analysis and visualization should be interpreted *given* the model performance (which is indicated by horizontal line on the Figures 4.10 and 4.11 which indicates cross-validated pooled testing RMSE; see Table 4.16). I will utilize this method with a better model in the [Direct and indirect effect, covariates and then some](#) section that has much lower cross-validated RMSE. What is important to remember with this analysis is that athletes who showed lower or higher observation compared to what was predicted by the model are flagged with red or green color. As opposed to the observed outcome approach, a model-based prediction approach uses *ceteris paribus* in estimating responders vs. non-responders, or at least providing residuals for such a decision. For example, everything else being equal, based on predictor variables, the

expected observation is higher or lower than the model prediction. This indicates that there might be something not-identified by predictive algorithm and thus needs to be flagged for a further analysis. But more about this in the [Direct and indirect effect, covariates and then some](#).

Besides analyzing residuals in the training data-set, we can also check how model predicts for each individual within cross-validation using Bias-Variance decomposition (see [Bias-Variance decomposition and trade-off](#) section). Figure 4.12 depicts prediction error (decomposed to Bias and Variance) for each athlete using ANCOVA RCT model.

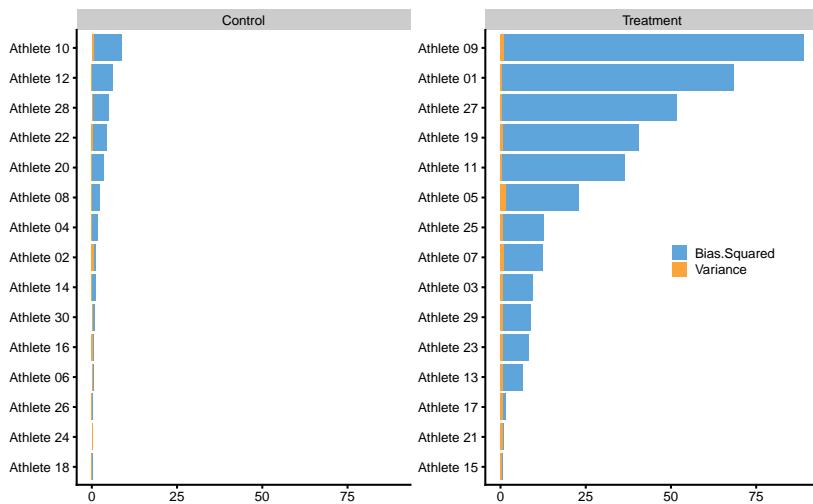


Figure 4.12: **Bias-variance across 10 times repeated 3-fold cross-validation for each athlete.** This analysis can also be utilized to flag certain observations (i.e. athlete in this case) that are troublesome for the predictive model

These graphs will make much more sense when more predictive model is applied in the [Direct and indirect effect, covariates and then some](#) section.

4.7.2 Counterfactual analysis and Individual Treatment Effects

As explained, causal inference and explanatory modeling aim to understand, or at least to quantify the causal (or treatment) effects. This is done by elaborate and transparent control of the confounders and study designs. Predictive modelling on the other hand is interested in providing valid predictions on new or unseen data without assuming underlying DGP by treating it as a black-box. In certain scenarios, when confounders are controlled, predictive modelling can be

interpreted causally (Zhao and Hastie 2019). With observational studies, there is always a risk of not controlling all important confounders, but transparency of the causal model is there to be falsified and discussed openly (Gelman and Hennig 2017; Hernan 2002; Hernán 2018, 2016; Hernán and Taubman 2008; Hernán, Hsu, and Healy 2019). Even with the RCTs, there might be uncertainties in applying the findings from the experiments to realistic settings (Gelman 2011; Heckman 2005).

PDP and ICE plots are *model-agnostic* tools for interpreting black-box models that can be used in causal interpretations only after the effort to define the causal structure with domain-specific expertise is taken into account. This approach can be used to estimate, based on the predictive model, counterfactual change and Post-test scores when athletes do not receive treatment (for the Treatment group), or when athletes receive treatment (for the Control group). This way potential outcomes are predicted.

To explain how this is done, let's again consider the Table 4.13. Changing the Group column (in this case `groupTreatment`) for every observation (athletes in this case) while keeping all other variables (i.e. columns) the same, we can get a glimpse into causal effects of the treatment (assuming the model and the data are valid for such an inference). The prediction model utilized is ANCOVA RCT model.

In the Table 4.18, columns `Post-test_0 (cm)` and `Post-test_1 (cm)` indicate these counterfactual changes for which we are interested how the model predicts. These predictions are in the columns `Post-test_0 (cm)` and `Post-test_1 (cm)`.

If we depict these changes in the Group for every athlete, we will get the ICE graph. Average of these predictions gives us the PDP graph. Figure 4.13 depicts PDP and ICE for the Group variable. We will get back to this graph in the [Direct and indirect effect, covariates and then some](#) section.

Table 4.18: Counterfactual table used to check how the model predicts when Group changes

Athlete	groupTreatment	Pre-test (cm)	Post-test (cm)	Change (cm)	groupTreatment_0	Post-test_0 (cm)	groupTreatment_1	Post-test_1 (cm)
Athlete 12	0	42.56	44.29	1.01	0	42.04	1	49.89
Athlete 28	0	47.06	47.98	0.55	0	46.07	1	53.92
Athlete 04	0	44.53	45.13	0.12	0	43.80	1	51.65
Athlete 02	0	49.63	48.86	-0.01	0	48.37	1	56.22
Athlete 08	0	41.11	42.13	-0.45	0	40.74	1	48.59
Athlete 26	0	42.31	41.61	-0.51	0	41.82	1	49.67
Athlete 06	0	45.96	45.70	-0.52	0	45.09	1	52.93
Athlete 14	0	44.51	42.89	-0.63	0	43.79	1	51.64
Athlete 18	0	42.57	42.15	-0.74	0	42.06	1	49.90
Athlete 16	0	37.63	37.83	-0.75	0	37.63	1	45.47
Athlete 22	0	36.52	34.83	-0.97	0	36.63	1	44.48
Athlete 24	0	40.15	39.88	-0.13	0	39.88	1	47.73
Athlete 30	0	39.34	38.34	-1.21	0	39.16	1	47.01
Athlete 20	0	38.94	36.97	-1.72	0	38.80	1	46.65
Athlete 10	0	36.77	34.15	-2.26	0	36.85	1	44.70
Athlete 01	1	37.98	52.86	14.83	0	37.94	1	45.79
Athlete 27	1	44.79	58.50	13.43	0	44.04	1	51.88
Athlete 19	1	46.77	59.76	12.99	0	45.81	1	53.66
Athlete 25	1	38.90	49.58	10.81	0	38.76	1	46.61
Athlete 03	1	41.29	51.41	10.34	0	40.90	1	48.75
Athlete 23	1	48.41	57.57	8.58	0	47.28	1	55.13
Athlete 17	1	44.81	51.41	7.85	0	44.06	1	51.91
Athlete 21	1	37.14	44.95	7.37	0	37.19	1	45.04
Athlete 15	1	46.69	52.73	6.14	0	45.75	1	53.59
Athlete 29	1	42.77	47.38	5.02	0	42.23	1	50.08
Athlete 13	1	49.66	54.11	4.46	0	48.40	1	56.25
Athlete 05	1	37.92	41.63	3.78	0	37.89	1	45.74
Athlete 07	1	41.03	45.41	3.42	0	40.67	1	48.52
Athlete 11	1	45.27	46.72	1.82	0	44.47	1	52.31
Athlete 09	1	47.61	45.62	-1.57	0	46.57	1	54.42

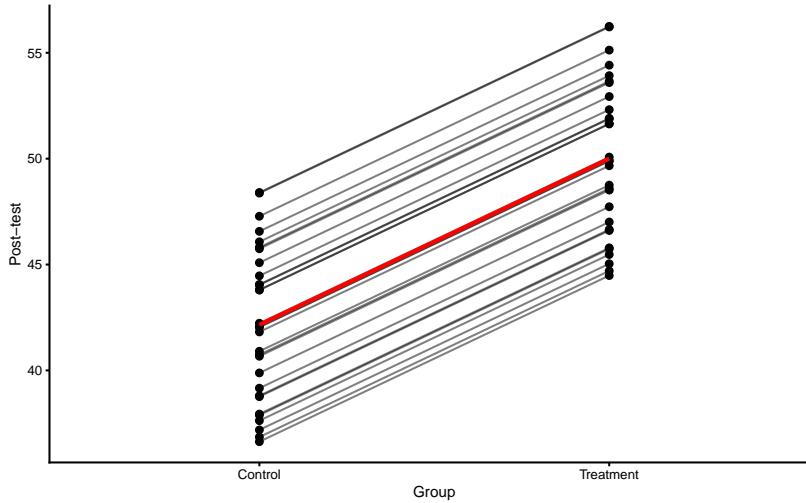


Figure 4.13: PDP and ICE plot for Group variable using ANCOVA RCT model

Besides PDP and ICE plot, we can also create a counterfactual plot for each athlete. For example, for the athletes in the Treatment group, we are interested how would the predicted Post-test *change* (given model used) if they are in the Control group and *vice versa* for the athletes from the Control group. This is

done by *flipping* “Treatment” and “Control” in the Group column and predicting Post-test using the trained model. Figure 4.14 depicts this visually for each athlete. Arrows represents predicted Change when *flipping* the Group variable.

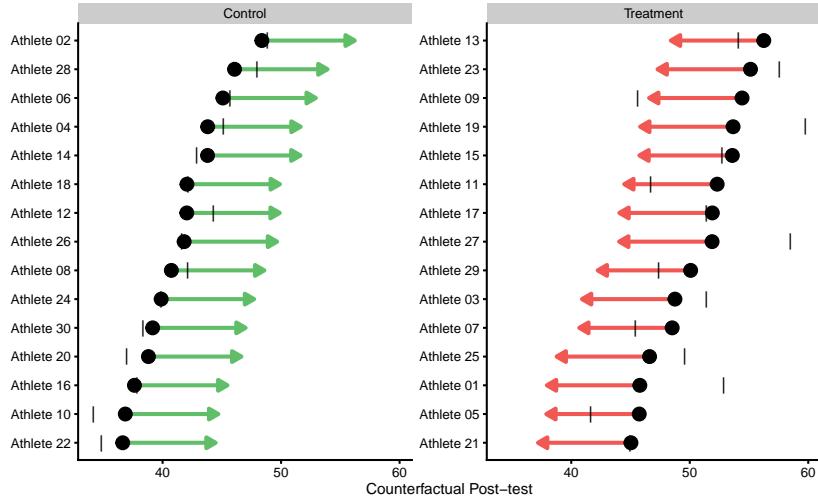


Figure 4.14: Individual counterfactual prediction when the Group changes. This way we can estimate model counterfactual predictions when the treatment changes (i.e. Controls receive treatment, and Treatment doesn’t receive treatment). Arrows thus represent model predicted *Individual Treatment Effects* (pITE). Arrows are color coded based on the magnitude of the effect using defined SESOI ($\pm 2.5\text{cm}$ in this example): grey for equivalent magnitude, green for lower magnitude, and red for higher magnitude. Vertical line indicates observed Post-test scores.

This analysis allows us to estimate counterfactual *individual causal (treatment) effects* (ITE) predicted by the model. These are indicated with the arrows on the Figure 4.14. Mathematically, arrow width is calculated using the Equation (4.8).

$$\widehat{ITE}_i = \widehat{y_i^{Group=Treatment}} - \widehat{y_i^{Group=Control}} \quad (4.8)$$

Since ANCOVA RCT model is used, which predicts average Group effect for every participant, estimated counterfactual ITEs are all the same and are equal to -7.85cm . Table 4.19 contains all individual model predictions using ANCOVA RCT model.

PDP and ICE plots, as well as individual treatment effects plots (and estimates) can be very valuable tool in visualizing the causal effects, which are appropriate in this case since we are analyzing RCT data. But we need to be very wary when using them with the observational data and giving them causal interpretation.

Table 4.19: Individual model predictions using ANCOVA RCT model

subject	group	observed	predicted	residual	magnitude	counterfactual	pITE	pITE_magnitude
Athlete 12	Control	44.29	42.04	-2.25	Equivalent	49.89	7.85	Higher
Athlete 28	Control	47.98	46.07	-1.91	Equivalent	53.92	7.85	Higher
Athlete 04	Control	45.13	43.80	-1.33	Equivalent	51.65	7.85	Higher
Athlete 02	Control	48.86	48.37	-0.48	Equivalent	56.22	7.85	Higher
Athlete 08	Control	42.13	40.74	-1.39	Equivalent	48.59	7.85	Higher
Athlete 26	Control	41.61	41.82	0.21	Equivalent	49.67	7.85	Higher
Athlete 06	Control	45.70	45.09	-0.61	Equivalent	52.93	7.85	Higher
Athlete 14	Control	42.89	43.79	0.90	Equivalent	51.64	7.85	Higher
Athlete 18	Control	42.15	42.06	-0.10	Equivalent	49.90	7.85	Higher
Athlete 16	Control	37.83	37.63	-0.21	Equivalent	45.47	7.85	Higher
Athlete 22	Control	34.83	36.63	1.80	Equivalent	44.48	7.85	Higher
Athlete 24	Control	39.88	39.88	0.00	Equivalent	47.73	7.85	Higher
Athlete 30	Control	38.34	39.16	0.82	Equivalent	47.01	7.85	Higher
Athlete 20	Control	36.97	38.80	1.84	Equivalent	46.65	7.85	Higher
Athlete 10	Control	34.15	36.85	2.70	Higher	44.70	7.85	Higher
Athlete 01	Treatment	52.86	45.79	-7.08	Lower	37.94	-7.85	Lower
Athlete 27	Treatment	58.50	51.88	-6.62	Lower	44.04	-7.85	Lower
Athlete 19	Treatment	59.76	53.66	-6.10	Lower	45.81	-7.85	Lower
Athlete 25	Treatment	49.58	46.61	-2.98	Lower	38.76	-7.85	Lower
Athlete 03	Treatment	51.41	48.75	-2.66	Lower	40.90	-7.85	Lower
Athlete 23	Treatment	57.57	55.13	-2.44	Equivalent	47.28	-7.85	Lower
Athlete 17	Treatment	51.41	51.91	0.49	Equivalent	44.06	-7.85	Lower
Athlete 21	Treatment	44.95	45.04	0.09	Equivalent	37.19	-7.85	Lower
Athlete 15	Treatment	52.73	53.59	0.86	Equivalent	45.75	-7.85	Lower
Athlete 29	Treatment	47.38	50.08	2.69	Higher	42.23	-7.85	Lower
Athlete 13	Treatment	54.11	56.25	2.14	Equivalent	48.40	-7.85	Lower
Athlete 05	Treatment	41.63	45.74	4.11	Higher	37.89	-7.85	Lower
Athlete 07	Treatment	45.41	48.52	3.11	Higher	40.67	-7.85	Lower
Athlete 11	Treatment	46.72	52.31	5.59	Higher	44.47	-7.85	Lower
Athlete 09	Treatment	45.62	54.42	8.80	Higher	46.57	-7.85	Lower

4.7.3 Direct and indirect effect, covariates and then some

In the previous RCT example, we have assumed *binary* treatment (either plyometric training is done or not), whereas in real life there can be nuances in the treatment, particularly in volume of jumps performed, making the treatment continuous rather than binary variable. This way, we are interested in the effects of number of jumps on the changes in vertical jump height.

There could also be *hidden variables* involved that *moderate* and *mediate* the effects of the treatment⁵. For example, the higher someone jumps in the Pre-test, the lower the change in the Post-test (i.e. it is harder to improve vertical jump height). Or, the stronger someone is in the Pre-test (measured using relative back squat 1RM) the more potentiated the effects of the plyometrics are. All these are needed expert subject-matter knowledge, required to understand the underlying DGP (and thus to avoid introducing bias in causal analyses; see Lübke et al. (2020)). With such causal structure, we do not have *direct treatment effect* (plyometric → change in vertical jump) only anymore, but moderated and mediated, or *indirect effects* estimated using the *interactions* in the regression models.

To explain these concepts, let's assume that that besides Pre-test and Post-test scores in our RCT study, we have also measured Back squat relative 1RMs since we believed that strength of the individual will moderate the effects of the plyometric treatment. This data is enlisted in the Table 4.20. Squat 1RM in this case represent characteristic of the subject, or a *covariate*. Additional covariates (not considered here) might include gender, experience, height, weight and so forth.

Since the individual are randomized into Treatment and Control groups, we expect that there is no difference between Squat 1RM between them. Figure 4.15 demonstrates that there is no difference between groups.

⁵Using the randomization in the RCT it is assumed that these hidden variables are equally distributed, and that there is no *selection bias* involved.

Table 4.20: Randomized control trial data but now with 1RM strength covariate

Athlete	Squat 1RM	Group	Pre-test (cm)	Post-test (cm)	Change (cm)
Athlete 12	1.53	Control	42.56	44.29	1.01
Athlete 28	1.59	Control	47.06	47.98	0.55
Athlete 04	1.49	Control	44.53	45.13	0.12
Athlete 02	1.31	Control	49.63	48.86	-0.01
Athlete 08	1.77	Control	41.11	42.13	-0.45
Athlete 26	1.34	Control	42.31	41.61	-0.51
Athlete 06	1.33	Control	45.96	45.70	-0.52
Athlete 14	2.03	Control	44.51	42.89	-0.63
Athlete 18	1.21	Control	42.57	42.15	-0.74
Athlete 16	1.91	Control	37.63	37.83	-0.75
Athlete 22	1.49	Control	36.52	34.83	-0.97
Athlete 24	1.37	Control	40.15	39.88	-0.27
Athlete 30	1.04	Control	39.34	38.34	-1.21
Athlete 20	1.58	Control	38.94	36.97	-1.72
Athlete 10	1.67	Control	36.77	34.15	-2.26
Athlete 01	2.05	Treatment	37.98	52.86	14.93
Athlete 27	2.05	Treatment	44.79	58.50	13.43
Athlete 19	1.87	Treatment	46.77	59.76	12.99
Athlete 25	1.97	Treatment	38.90	49.58	10.81
Athlete 03	1.79	Treatment	41.29	51.41	10.34
Athlete 23	1.44	Treatment	48.41	57.57	8.58
Athlete 17	1.21	Treatment	44.81	51.41	7.85
Athlete 21	1.49	Treatment	37.14	44.95	7.37
Athlete 15	1.43	Treatment	46.69	52.73	6.14
Athlete 29	1.22	Treatment	42.77	47.38	5.02
Athlete 13	1.18	Treatment	49.66	54.11	4.46
Athlete 05	1.15	Treatment	37.92	41.63	3.78
Athlete 07	1.21	Treatment	41.03	45.41	3.42
Athlete 11	0.86	Treatment	45.27	46.72	1.82
Athlete 09	0.69	Treatment	47.61	45.62	-1.57

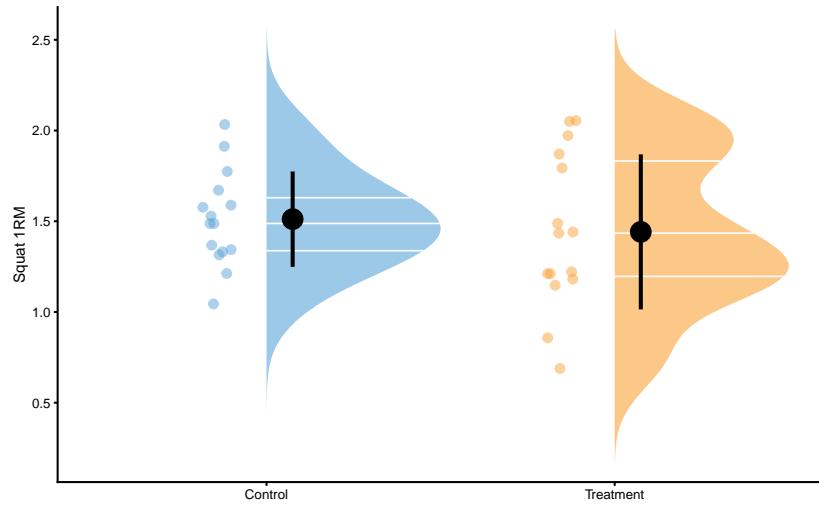


Figure 4.15: **Squat 1RM scores for Control and Treatment groups.**
Since athletes are randomized, we expect them to be similar (i.e. no selection bias involved)

Before modeling this RCT data, let's check visually the relationship between Pre-test and Change (panel A), Squat 1RM and Change (panel B), and Pre-test and Squat 1RM (panel C) (Figure 4.16).

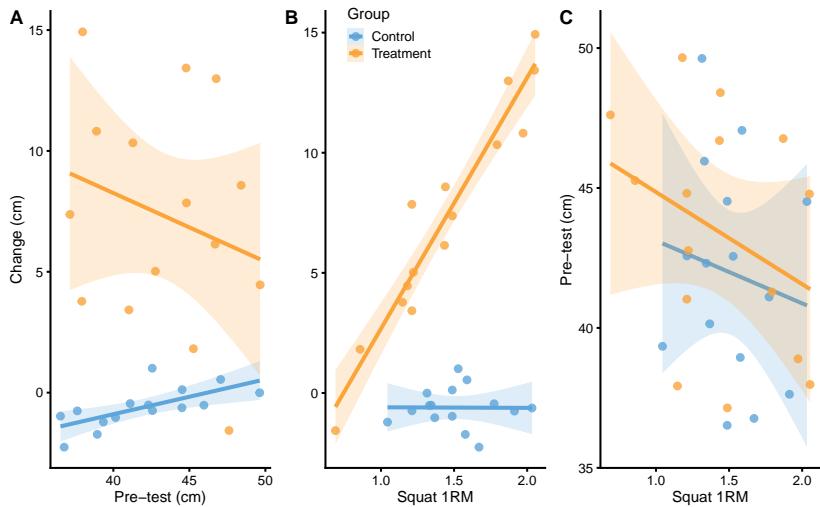


Figure 4.16: **Relationship between predictors.** **A.** Relationship between Pre-test and Change scores. Lines indicate linear regression model and are used to indicate relationship and interaction. Colored area represent *95% confidence intervals* (see [Statistical inference](#) section for more information). If there is no interaction between Pre-test and Change score, the lines would be parallel. As can be seen with the Treatment group, it seems that individuals with higher Pre-test demonstrates lower Change, and *vice versa* for the Control group. But if we check the confidence interval area, these are due to random chance. **B.** Relationship between Squat 1RM and Change scores. Visual analysis indicates that individuals in the Treatment group with the higher Squat 1RM demonstrates higher changes after the training intervention. **C.** Relationship between Squat 1RM and Pre-test predictors. It seems that, in both groups, athletes with higher Squat 1RM scores have lower Pre-test values. Checking confidence interval areas, this relationship is due to random chance.

As can be seen from the Figure 4.16, there seems to be some *interaction* between Squat 1RM and Change. Other panels indicate that there might be some relationship (i.e. *interaction*), but *95% confidence intervals* around the regression lines indicate that these are due to *random chance* (see [Statistical inference](#) chapter for more info; for now you do not need to understand this concept).

What does interaction mean? If we check the panel C in the Figure 4.16, interaction refers to change in regression line slopes. If the Group regression lines are parallel, then the distance between them is due to effect of treatment (i.e. direct or *main* effect of treatment). Since they are not parallel, it means that Squat 1RM and treatment interact: the higher the Squat 1RM for the Treatment group, the higher the Change, while that is not the case for the Control group. Mathematically, interaction is a simple multiplication between two predictors, as can be seen in the Table 4.21.

Table 4.21: RCT data with interaction between Group and Squat 1RM

Athlete	groupTreatment	Squat 1RM	groupTreatment:Squat 1RM	Pre-test (cm)	Change (cm)
Athlete 12	0	1.53	0.00	42.56	1.01
Athlete 28	0	1.59	0.00	47.06	0.55
Athlete 04	0	1.49	0.00	44.53	0.12
Athlete 02	0	1.31	0.00	49.63	-0.01
Athlete 08	0	1.77	0.00	41.11	-0.45
Athlete 26	0	1.34	0.00	42.31	-0.51
Athlete 06	0	1.33	0.00	45.96	-0.52
Athlete 14	0	2.03	0.00	44.51	-0.63
Athlete 18	0	1.21	0.00	42.57	-0.74
Athlete 16	0	1.91	0.00	37.63	-0.75
Athlete 22	0	1.49	0.00	36.52	-0.97
Athlete 24	0	1.37	0.00	40.15	-1.03
Athlete 30	0	1.04	0.00	39.34	-1.21
Athlete 20	0	1.58	0.00	38.94	-1.72
Athlete 10	0	1.67	0.00	36.77	-2.26
Athlete 01	1	2.05	2.05	37.98	14.93
Athlete 27	1	2.05	2.05	44.79	13.43
Athlete 19	1	1.87	1.87	46.77	12.99
Athlete 25	1	1.97	1.97	38.90	10.81
Athlete 03	1	1.79	1.79	41.29	10.34
Athlete 23	1	1.44	1.44	48.41	8.58
Athlete 17	1	1.21	1.21	44.81	7.85
Athlete 21	1	1.49	1.49	37.14	7.37
Athlete 15	1	1.43	1.43	46.69	6.14
Athlete 29	1	1.22	1.22	42.77	5.02
Athlete 13	1	1.18	1.18	49.66	4.46
Athlete 05	1	1.15	1.15	37.92	3.78
Athlete 07	1	1.21	1.21	41.03	3.42
Athlete 11	1	0.86	0.86	45.27	1.82
Athlete 09	1	0.69	0.69	47.61	-1.57

But as already alluded, the use of Change scores should be avoided. Let's see this exact graphs, but using Post-test (Figure 4.17) as our variable of interest (i.e. outcome variable).

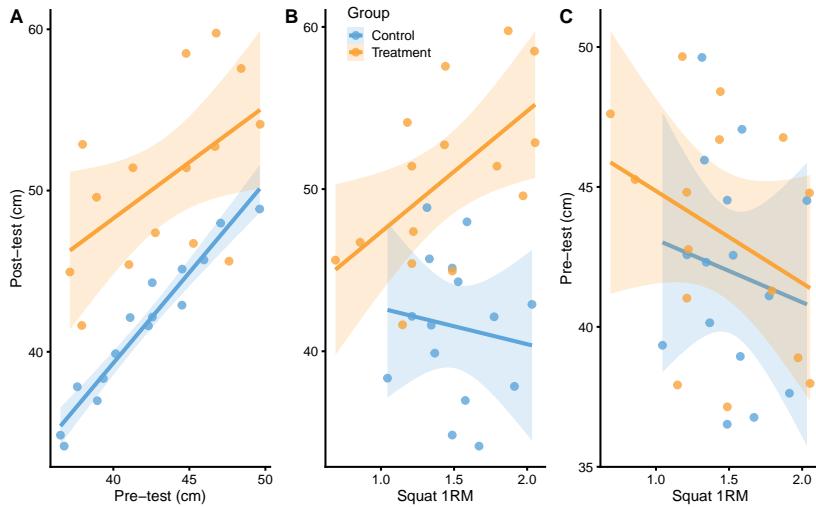


Figure 4.17: Relationship between predictors. **A.** Relationship between Pre-test and Post-test scores. Lines indicate linear regression model and are used to indicate relationship and interaction. Colored area represent *95% confidence intervals* (see Statistical inference section for more information). If there is no interaction between Pre-test and Post-test score, the lines would be parallel. As can be seen on the figure, there doesn't seem to be interaction involved. **B.** Relationship between Squat 1RM and Post-test scores. Visual analysis indicates that individuals in the Treatment group with the higher Squat 1RM demonstrates higher Post-test scores after the training intervention. **C.** Relationship between Squat 1RM and Pre-test predictors. It seems that, in both groups, athletes with higher Squat 1RM scores have lower Pre-test values. Checking confidence interval areas, this relationship is due to random chance.

Let's apply linear regression model to these predictors. The parameters we are going to estimate are enlisted in the linear Equation (4.9).

$$\widehat{Post} = \hat{\beta}_0 + \hat{\beta}_1 Group + \hat{\beta}_2 Pre + \hat{\beta}_3 Squat\ 1RM + \hat{\beta}_4 Group : Squat\ 1RM \quad (4.9)$$

Estimated parameters for this linear model with interaction are enlisted in the Table 4.22.

If we check the estimated coefficient for the group Treatment in the Table 4.22, which is equal to -8.03, can we conclude that this is the whole effect of

Table 4.22: Estimated linear regression parameters for the RCT model with interaction between Group and Squat 1RM

Intercept	groupTreatment	Pre-test	Squat 1RM	Group:Squat 1RM
-5.21	-8.03	1.1	0.26	10.82

treatment (i.e. plyometric treatment)? Luckily no! This coefficient represents *direct treatment effect*, but there are *indirect treatment effects* due to Squat 1RM and interaction between Squat 1RM and Group. This also assumes that we have not introduced bias in treatment effect estimation by *adjusting* (or by *not adjusting*) for covariates that we should not adjust for (or *vice versa*; for great applied paper please refer to Lübke et al. (2020)).

Figure 4.18 depicts residuals of the RCT model with interactions.

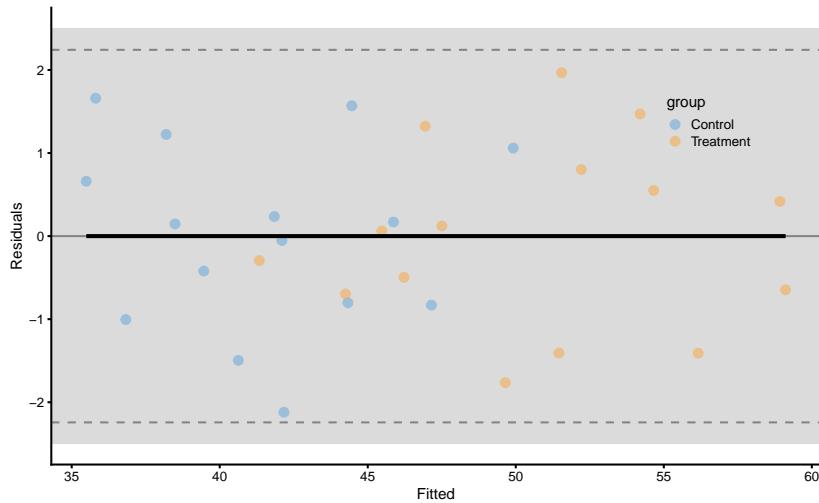


Figure 4.18: Residuals of the linear regression RCT model with interaction. Grey band on both panels represents SESOI of $\pm 2.5\text{cm}$

As opposed to the Figures 4.6 and 4.8, we can quickly see that this model have almost all residuals in the SESOI band. Table 4.23 contains cross-validated model performance (using the same 10 repeats of 3 folds cross-validation splits as used in the simple and ANCOVA RCT models).

As expected, predictive performance metrics are now much better. Let's inspect the athlete's residuals (Figure 4.19).

Table 4.23: Cross-validated predictive performance metrics for the RCT model with interaction

metric	training	training.pooled	testing.pooled	mean	SD	min	max
MBE	0.00	0.00	0.07	0.07	0.61	-1.40	1.18
MAE	0.90	0.81	1.19	1.19	0.25	0.75	1.76
RMSE	1.08	1.00	1.46	1.43	0.27	0.89	1.99
PPER	0.97	0.99	0.91	0.87	0.06	0.74	0.97
SESOI to RMSE	4.64	5.02	3.43	3.61	0.73	2.52	5.61
R-squared	0.97	0.98	0.95	0.95	0.02	0.88	0.98
MinErr	-2.12	-2.40	-2.72	-1.90	0.58	-2.72	-0.76
MaxErr	1.97	2.12	3.15	2.27	0.80	0.32	3.15
MaxAbsErr	2.12	2.40	3.15	2.57	0.42	1.65	3.15

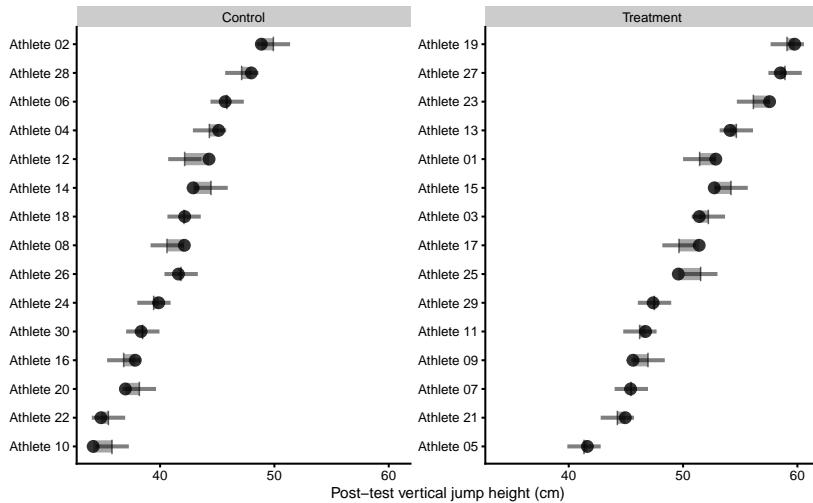


Figure 4.19: Observed Post-test in vertical jump for each athlete in the study. Black dot indicate observed Post-test; vertical line indicate model prediction; colored bar indicated the residual magnitude compared to defined SESOI ($\pm 2.5\text{cm}$ in this example): grey for equivalent magnitude, green for lower magnitude, and red for higher magnitude; horizontal error bar represents cross-validated RMSE (see Table 4.23, RMSE metric, column *testing.pooled*) and is used to indicate visually model predictive performance and uncertainty around model prediction

If we compare residuals from the simple model (Figure 4.11) and ANCOVA RCT model (Figure 4.10), in RCT model with interactions the residuals are much smaller, indicating better prediction. You can also see that model predictions differ, as opposed to simple model that predicted same Change values for all athletes in the Control and Treatment groups. Athletes who are flagged (have

residual bigger than SESOI) might need further inspection, since given training data and model, these demonstrate Post-test that is larger/smaller than expected taking their covariates into account (in this case Strength 1RM).

Figure 4.20 depicts prediction errors during cross-validation for each athlete. This analysis, together with the Figure 4.19, can be used to detect athletes that are *troublesome* for the predictive model, and thus bear some further inspection.

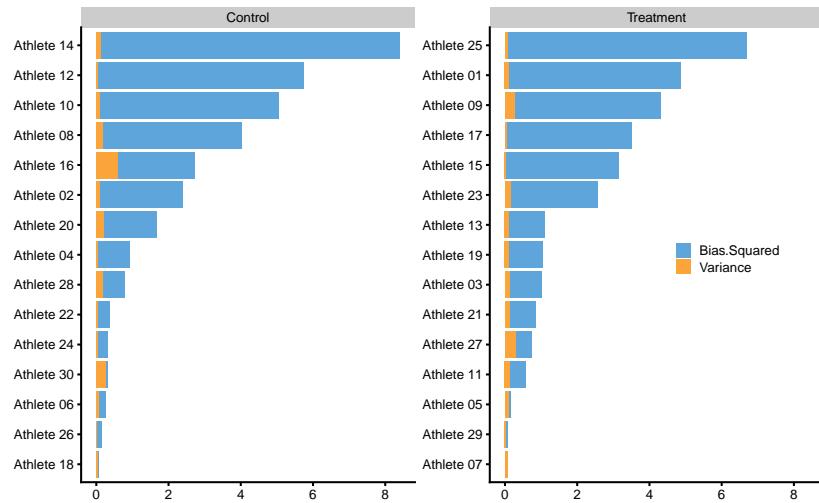


Figure 4.20: **Bias-variance across 10 times repeated 3-fold cross-validation for each athlete.** This analysis can also be utilized to flag certain observations (i.e. athlete in this case) that are troublesome for the predictive model

Interpreting and understanding *direct* and *indirect* effects can be quite difficult, especially when causal structure becomes complex. Visualization techniques such as already mentioned PDP and ICE graphs can be utilized to understand the causal mechanism (Zhao and Hastie 2019; Goldstein et al. 2013). These techniques can also be implemented in observational studies, but with considerable domain knowledge and assumptions needed (Zhao and Hastie 2019). Although predictive analysis, particularly those using *black box* machine learning models, has been criticized to lack causal interpretation (Hernán, Hsu, and Healy 2019; Pearl and Mackenzie 2018; Pearl 2019), they can complement causal (or explanatory) analysis (Breiman 2001; Shmueli 2010; Yarkoni and Westfall 2017). Figure 4.21 depicts PDP and ICE plots for Group and Strength 1RM predictors.

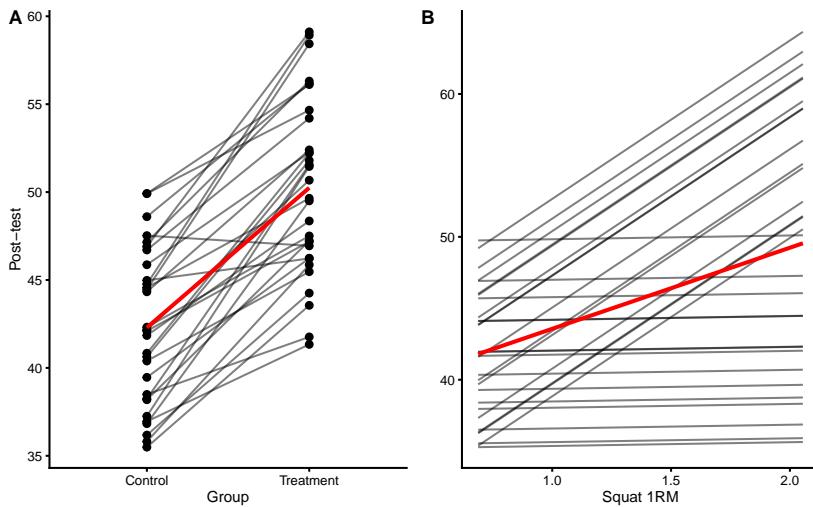


Figure 4.21: **PDP and ICE plots for Group and Strength 1RM predictors using RCT model with interaction**

Since this is RCT, we can give causal interpretation to PDP and ICE plot, particularly panel B in the Figure 4.21. According to this analysis (given the data and the model), if one increase Squat 1RM, the effect of treatment (i.e. plyometric training) will be higher. This can be further analyzed using each athlete. The question we would like to answer (given the data collected and the model) is “How would particular athlete respond to the treatment if his strength was higher or lower?”. Figure 4.22 shows ICE plots in separate facets. This gives us the ability to analyze (given the data and the model) how would each athlete respond if his or her Squat 1RM changed.

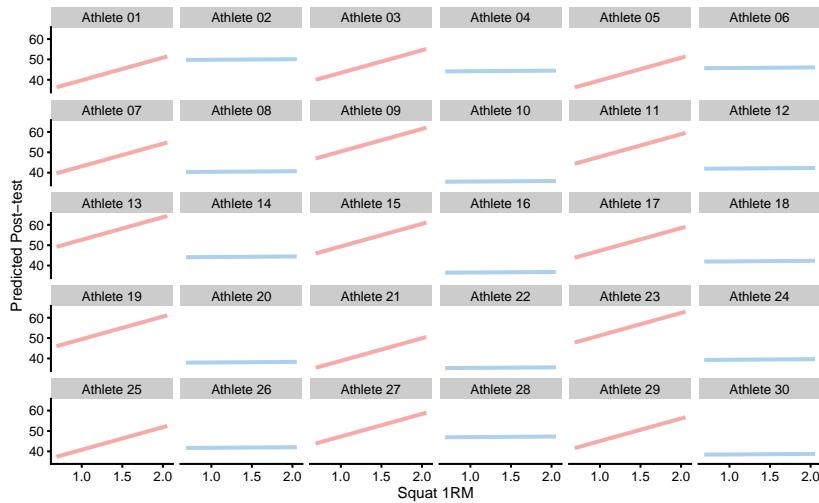


Figure 4.22: ICE plots for each athlete

Figure 4.23 depicts predicted ITE (i.e. what would happen if Groups *flipped* and everything else being equal). If we compare Figure 4.23 with Figure 4.14, we can quickly see that the ITEs differ and are not equal for every individual. If we calculate the mean of the ITEs (i.e. arrow lengths), we will get an estimate of ATE. SD of ITEs will give us estimate how variable the treatment effects are, or estimate of VTE. Since these are predicted with the model, I've used the terms pATE and pVTE indicating that they are estimated with the predictive model.

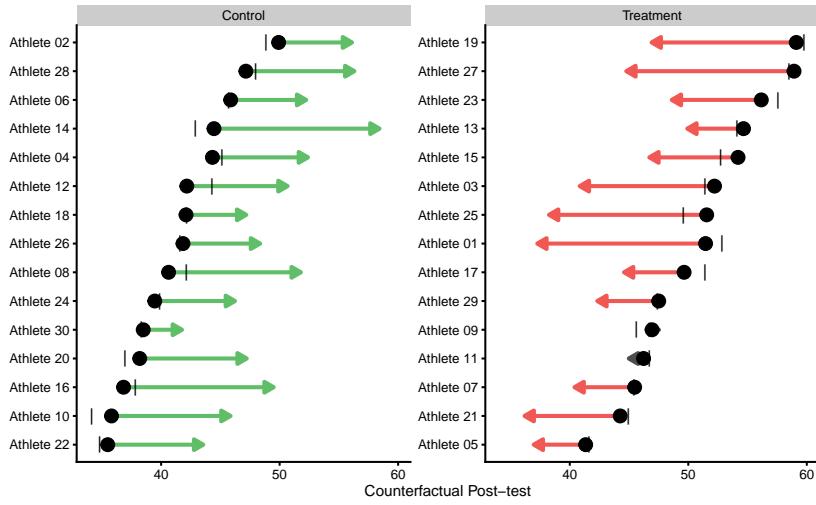


Figure 4.23: **Individual counterfactual prediction when the Group changes.** This way we can estimate model counterfactual predictions when the treatment changes (i.e. Controls receive treatment, and Treatment doesn't receive treatment). Arrows thus represent model predicted *Individual Treatment Effects* (ITE). Vertical line indicates observed Change

Table 4.24 contains comparison between the ANCOVA RCT model ($\widehat{Post} = \hat{\beta}_0 + \hat{\beta}_1 Group + \hat{\beta}_2 Pre$) and model with Squat 1RM and interaction term ($\widehat{Post} = \hat{\beta}_0 + \hat{\beta}_1 Group + \hat{\beta}_2 Pre + \hat{\beta}_3 Squat 1RM + \hat{\beta}_4 Group : Squat 1RM$) for few estimators that are insightful for the comparison.

As can be seen in the Table 4.24, linear model with interactions predicts Post-test in the vertical jump height better than the ANCOVA model. This means that the variance in the treatment effect (SD Residual in the Table 4.24) is now explained with additional variables and their interactions⁶.

We are not only interested in prediction, but rather in the underlying causal structure and explaining this model performance when we *intervene* on variables (i.e. what happens to the target variable when I change X from *a* to *b*, while keeping other variables fixed - which is *ceteris paribus* condition). For this reason, PDP and ICE plots can be used to give causal interpretation of the model (see Figure 4.21). The main critique of Judea Pearl regarding the use of predictive models and techniques is the lack of counterfactual causal interpretation (Pearl

⁶If you remember the footnote discussion on aleatory and epistemic uncertainty, this is example where what we believed to be aleatory uncertainty (inter-individual residual error or variation that we could not predict: SD Residual in the Table 4.24) was actually epistemic uncertainty that we reduced with introducing new variables (Squat 1RM and interaction term) to the model. Although the inter-individual variation in the treatment effect remains (SD_{TE} or SD_{IR}), we are now able to predict individual responses more accurately.

Table 4.24: Comparison between two models using few estimators and performance metrics

estimator	ANCOVA.model	Interaction.model
Training RMSE	3.31	1.08
Training PPER	0.54	0.97
Training SESOI to RMSE	1.51	4.64
Training R-squared	0.75	0.97
CV RMSE	3.68	1.46
CV PPER	0.50	0.91
CV SESOI to RMSE	1.36	3.43
CV R-squared	0.69	0.95
SD Residual (Treatment)	1.43	1.12
SD Residual (Control)	4.64	1.11
pATE (Treatment)	-7.85	-7.57
pATE (Control)	7.85	8.33
pVTE (Treatment)	0.00	4.62
pVTE (Control)	0.00	2.84

and Mackenzie 2018; Pearl 2019), particularly with observational studies. I agree that “the role of expert knowledge is the key difference between prediction and causal inference tasks” (Hernán, Hsu, and Healy 2019, 44) and that “both prediction and causal inference require expert knowledge to formulate the scientific question, but only causal inference requires causal expert knowledge to answer the question” (Hernán, Hsu, and Healy 2019, 45), but this doesn’t negate the need for providing predictive performance, as well as helping in interpreting the model when such data is available (Zhao and Hastie 2019).

According to Zhao and Hastie (Zhao and Hastie 2019, 1): “There are three requirements to make causal interpretations: a model with good predictive performance, some domain knowledge in the form of a causal diagram and suitable visualization tools.”. The common denominator among multiple experts is that for causal inference and causal interpretation there is a need for domain knowledge, particularly when RCTs are not available. This domain knowledge can be made more transparent by using DAGs and other structural diagrams, and thus help in falsifying assumptions (Gelman and Hennig 2017; Hernán 2017; Hernan 2002; Hernán, Hsu, and Healy 2019). Adding prediction to explanatory models can be seen as complement, particularly since statistical analysis has been neglecting predictive analysis over explanatory analysis (Breiman 2001; Shmueli 2010; Yarkoni and Westfall 2017).

4.7.4 Model selection

Besides applying model with a single interaction term, we can also apply models with more interactions, or quadratic or polynomial terms, or what have you. As we have seen in the [Prediction](#) chapter, these models can overfit quite easily. That is why we utilized cross-validation to check for model performance on the unseen data. But what if two or more models of different complexity perform similarly on the unseen data (i.e. when cross-validated)? This is the problem of *model selection and comparison*. Generally, we want to select the simplest model that gives us reliable predictions. Discussions regarding the model comparison and model selection are beyond the scope of this book, although I will provide few examples in the second part of the book.

The model definition should rely on pre-existing beliefs (i.e. subject-matter expert knowledge) around causal structure underlying intervention. If the statistical analysis is done to *confirm* the structural causal model, then this represents *confirmatory analysis*. Usually, these studies need to be *pre-registered* with the exact analysis workflow and assumption defined *before* data is collected. This is required because in the *exploratory analysis* one can *play* with the data, different models can be tried and the model that fits the data best can be selected. Exploratory analysis is useful for generating models and hypothesis for future studies, but also introduces *hindsight bias* since the model is selected *after* seeing the data or the results of multiple analyses. Very related is so called *p-harking* (Hypothesizing After Results are Known) which involves modifying the hypothesis or in this case causal model, after seeing the results, most often with the aim to reach *statistical significance* (discussed later in the [Statistical inference](#) section). In predictive analysis this hindsight bias is reduced by using hold-out sample, cross-validation, and evaluating the final model performance on the data that has not been seen by the model.

4.8 Ergodicity

Ergodic process is underlying DGP that is equivalent at *between-individual* (or inter-individual; or group-based analysis) and *within-individual* (or intra-individual or simply individual-based analysis) levels. Thus the causal effects identified using between-individual analysis can be applied to understand within-individual causation as well. *Non-Ergodic* process on the other hand differs between these two levels and effects identified at between-individual level cannot be inferred to within individual level.

Few authors have already brought into question the generalizability of group-based research to individual-based prediction and causal inferences (Fisher, Medaglia, and Jeronimus 2018; Glazier and Mehdizadeh 2018; Molenaar 2004; Molenaar and Campbell 2009). Here is an interesting quote from Molenaar (Molenaar and Campbell 2009, 112):

“Most research methodology in the behavioral sciences employs inter-individual analyses, which provide information about the state of affairs of the population. However, as shown by classical mathematical-statistical theorems (the ergodic theorems), such analyses do not provide information for, and cannot be applied at, the level of the individual, except on rare occasions when the processes of interest meet certain stringent conditions. When psychological processes violate these conditions, the inter-individual analyses that are now standardly applied have to be replaced by analysis of intra-individual variation in order to obtain valid results.”

The individual counterfactual predictions (ITEs) and ICE plots thus rely on ergodicity, which represents big assumption. This means that we should be cautious when generalizing model predictions and causal explanations from group-based level to individual-level.

Data analysis at the individual level (i.e. collecting multiple data point for individuals) and identifying individual causal effects and predictions might be the step forward, but even with such an approach we are *retrodicting* under *ceteris paribus* conditions and we might not be able to predict future responses. For example, if we have collected responses to various interventions for a particular individual, we might not be able to estimate with certainty how he or she is going to respond to a familiar treatment in the future, since such a prediction relies on *stationarity* of the parameters or the underlying DGP (Molenaar and Campbell 2009).

But this doesn't imply that all our efforts are useless. We just need to accept the uncertainty and the assumptions involved. For example, for completely novel subject, the best response estimate or prediction estimate is the expected response calculated by using the group-based analysis (i.e. average treatment effect). This represents the most likely response or prediction. But on top of providing these distribution-based or group-based estimates, one can provide expected uncertainties showing individual-based or response proportions as anchor-based (magnitude-based) estimates (Estrada, Ferrer, and Pardo 2019; Norman et al. 2001). It is important to note that these usually estimate the same information (Estrada, Ferrer, and Pardo 2019; Norman et al. 2001); e.g. the higher the Cohen's d the higher the proportion of higher responses. Thus reporting magnitude-based proportions as uncertainties together with expected responses using average-based approach at least help in communicating uncertainties much better than reporting solely average-based estimates. When warranted with the research question, researchers should also report predictive performances on unseen data, as well as underlying assumption using graphical models such as DAGs.

It might be the best to conclude the section on causal inference with the quote from Andrew Gelman's paper (Gelman 2011, 965):

“Casual inference will always be a challenge, partly because our psychological intuitions do not always match how the world works. We like to think of simple causal stories, but in the social world, causes and effects are complex. We—in the scientific community—still have not come to any agreement on how best to form consensus on causal questions: there is a general acceptance that experimentation is a gold standard, but it is not at clear how much experimentation should be done, to what extent we can trust inference from observational data, and to what extent experimentation should be more fully incorporated into daily practice (as suggested by some in the “evidence-based medicine” movement).”

Chapter 5

Statistical inference

Why did estimates diverge from the *true* parameters in the previous examples? Why did estimated treatment effect differed from the true treatment effect we used to generate the data? This brings us to the important concept in statistical analysis: a difference between the *true parameter* in the *population* and the *sample estimate*. Population refers to all members of specified group, whereas sample contains only a part, or a subset of a population from which it is taken. To generate the data in simulations, we have assumed DGP in the population, out of which a single or multiples samples were drawn.

The mathematical notation to differentiate true population parameters and sample estimates, or statistics involve using Greek letters to represent true population parameters, and Latin letters to represent sample estimates. For example, μ stands for true *population mean*, while \bar{x} stand for *sample mean*, where *bar symbol* (" $\bar{}$ ") indicates *mean*. When it comes to standard deviation, σ stands for true population parameter, while SD stands for sample estimate. Sometimes the *hat symbol* (" $\hat{}$ ") is used to denote estimator. For example, \hat{y}_i would be a model estimate or prediction of the observed y_i .

The difference between the true population parameter and the sample estimate (assuming correctly identified and represented DGP among other issues such as the use of non-biased estimators for example) is due to the *sampling error*, that we will covered shortly. In the simulations, we are certain about the true population parameters, but in *real life* we are *uncertain* about the true parameters and we need to somehow quantify this uncertainty. This is the objective of statistical inference. In other words, with statistical inference we want to generalize from sample to a population, while taking into account uncertainties of the estimates.

5.1 Two kinds of uncertainty, two kinds of probability, two kinds of statistical inference

In the previous RCT example, by using the Pre-Test score and Group variables, we have estimated predictive performance of the model using Post-test as the target variable. Error (or uncertainty) around individual Post-test score was much bigger than SESOI, which made individual prediction practically useless. This prediction uncertainty decreased as new variables were introduced to the prediction model. With model involving Squat 1RM variable, we have achieved much better predictive performance. This type of uncertainty can be called *epistemic uncertainty* (O'Hagan 2004), which is a result of lack of knowledge or incomplete information.

In the [Prediction](#) section of this book, I have utilized *irreducible error* in the DGP to represent stochastic component or the random error. Due to this random error, scores will differ from sample to sample. This type of uncertainty can be called *aleatory uncertainty* (O'Hagan 2004), which results due intrinsic randomness. Another flagship example of the aleatory uncertainty is tossing a dice, drawing a card from a shuffled pack, or random sampling that produces the sampling error.

You can argue of course that aleatory uncertainty is ultimately epistemic uncertainty. For example, if I knew infinite details about the dice tossing, I would be able to predict exactly the number that will be landed. Philosophers have been arguing about these issues for ages, and it is not in the domain of this book to delve deeper into the matters of uncertainty.

The theory of statistical inference and statistics in general rests on describing uncertainties by using *probability*. Since there are two kinds of uncertainty, there are two kinds of probabilities and their meaning. Aleatory uncertainties, like tossing a dice or random sampling, are described using *long-frequency* definition of probability. For example, it can happen that I toss six for 4 times in a row, but in the long-run, which means infinite number of times, probability of tossing a six is equal to 1/6, or probability of 0.166, or 16.6% (assuming a fair dice of course). Probability viewed from this perspective represents long-frequency number of occurrences of the event of interest, divided by number of total events. For example, if I toss a dice for 1000 times, and if I get number six for 170 times, the probability of tossing a six is equal to 170 / 1000, or 17%.

With epistemic uncertainty, probability of a proposition simply represents a *degree-of-belief* in the truth of that proposition (O'Hagan 2004). The degree-of-belief interpretation of probability is referred to as *subjective probability* or *personal probability*, while long-frequency interpretation of probability is referred to as *objective probability*. There are two major schools of statistical inference leaning either on long-frequency interpretation of probability, called *frequentist*, or leaning on degree-of-belief interpretation of probability, called *Bayesian*. There are of course many nuances and other schools of statistical inference (Dienes 2008; Efron and Hastie 2016; Gelman and Hennig 2017) which are beyond the

5.1. TWO KINDS OF UNCERTAINTY, TWO KINDS OF PROBABILITY, TWO KINDS OF STATISTICAL INFERENCE

scope of this book. The additional approach to inference that will be considered in this book as a preferred approach is the *bootstrap* (Efron and Hastie 2016; Hesterberg 2015; Guillaume A Rousselet, Pernet, and Wilcox 2019b, 2019a). But more about it later.

To better explain the differences between the frequentist and Bayesian approach to statistical inference I am going to use known male mean height and SD in the population. The population parameter of interest is the **population mean** (Greek μ) estimated using the **sample mean** (\bar{x}).

Chapter 6

Frequentist perspective

As already stated, using simulations is outstanding teaching tool (Carsey and Harden 2013; Hopkins 2007), and also very useful for understanding the frequentist inference as well. Figure 6.1 (Panels A and B) depicts hypothetical male population where mean height μ is 177.8cm and SD (σ) is 10.16cm. From this population we are randomly *drawing* N=5 (left side panels on Figure 6.1) and N=20 (right side panels on Figure 6.1) individuals for which we estimate the **mean** height. Individuals are represented as blue dots (Panels C and D on Figure 6.1), whereas estimated **mean** height is depicted as orange dot. Now imagine we repeat this sampling 50 times, calculate the mean for every sample, and then draw the distribution of the sampled means (Panels E an F on Figure 6.1). This distribution is called *sampling distribution of the sample mean* and the SD of this distribution is referred to as *standard error* or *sampling error*. Since our estimate of interest is the **mean**, standard deviation of the sampling distribution of the mean is called *standard error of the mean* (SEM). On Panels E and F in the Figure 6.1, mean of the sampling **means** is indicated by a black dot, and error bars represent SEM.

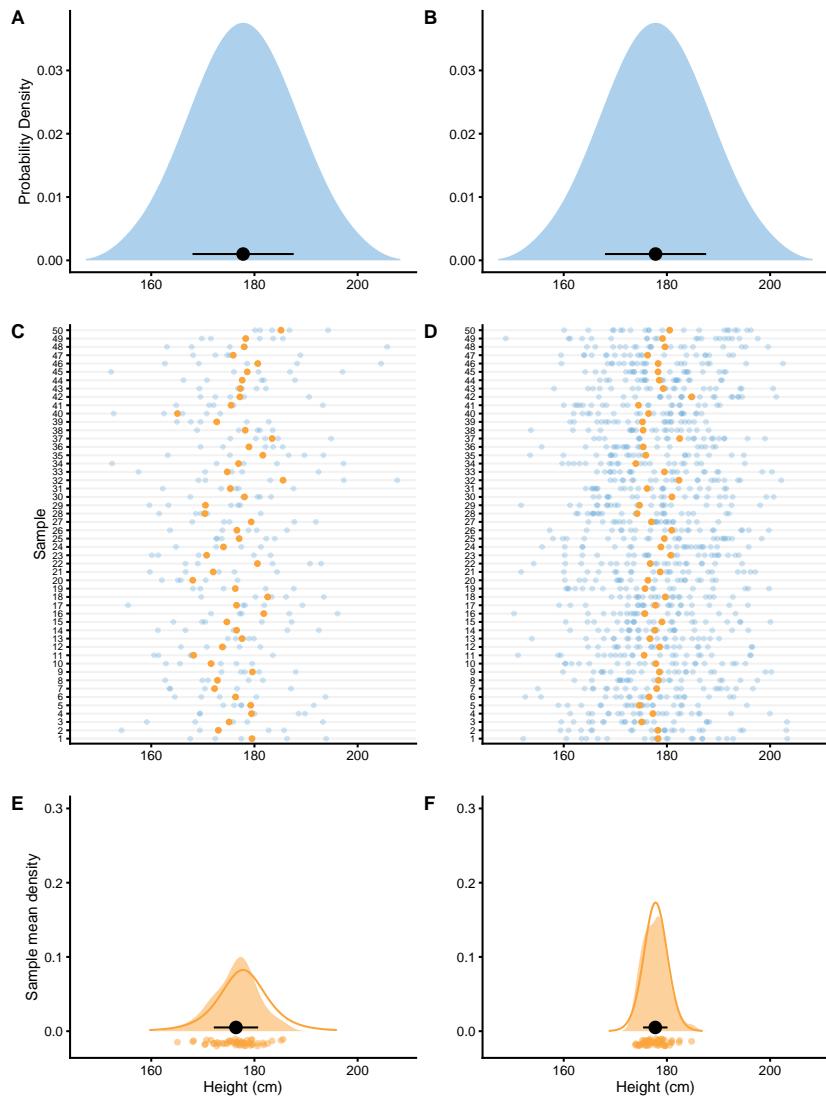


Figure 6.1: Sampling distribution of the mean. **A and B.** Distribution of the height in the population. From this population we draw samples. **C and D.** 50 sample are taken with $N=5$ (panel C) and $N=20$ (panel D) observations. Each observation is indicated by a blue dot. Calculated **mean**, as a parameter of interest, is indicated by an orange dot. **E and F.** Distribution of collected sample **means** from panels C and D. This distribution of the sample **means** is narrower, indicating higher precision when higher N is used. Black dot indicates the mean of the sample **means**, with error bars indicating SD of sample means. Orange line represents hypothetical distribution of the sample **means** when number of samples is infinitely large

As can be seen from the Figure 6.1, the sampling distribution of the `mean` looks like normal distribution. If the number of samples reach very large number or *infinity*, the sampling distribution of the `mean` will eventually be distributed with the `SEM` equal to (Equation (6.1)):

$$SEM = \frac{\sigma}{\sqrt{N}} \quad (6.1)$$

This *theoretical* distribution is overlaid on the acquired sampling distribution from 50 samples in the Figure 6.1 (Panels E and F). Since the true σ is not known, sample `SD` is utilized instead, in order to estimate the true `SEM` (Equation (6.2)):

$$\hat{SEM} = \frac{SD}{\sqrt{N}} \quad (6.2)$$

The take-home point is that the larger the sample, the smaller the standard error, which is visually seen as narrower sampling distribution (compare $N=5$ and $N=20$ sampling distributions on Figure 6.1). Another conclusion regarding frequentist inference, is that calculated probabilities revolve around sampling distribution of the sample `mean` and other long-frequency sampling distributions. Everything else are details. But as the saying goes, the devil is in the details.

Sampling distributions and equations for standard errors are derived algebraically for most estimators (e.g. `mean`, `SD`, Cohen's `d`), but for some estimators it might be hard to derive them, so other solutions do exist (like *bootstrapping* which will be covered in [Bootstrap](#) section). For example, sampling distribution of the change score proportions can be very difficult to be derived algebraically (Swinton et al. 2018). For some estimators, mean of the long-frequency samples is different than the true population value, thus these estimators are termed *biased estimators*. One example of the biased estimator would be `SD` of the sample where we divide with N , instead of $N - 1$. Estimators that have the mean of the long-frequency sample estimate equal to the true population parameter are called *unbiased estimators*.

Although the sampling distribution of the `mean` looks like a normal distribution, it actually belongs to the *Student's t* distribution, which has fatter tails for smaller samples (Figure 6.2). Besides `mean` and `SD`, Student's t distribution also has *degrees of freedom* (DF) parameters, which is equal to $N-1$ for the sample `mean`. Normal distribution is equal to Student's t distribution when DF is infinitely large.

Table 6.1: Critical values for Student's t distribution with different degrees of freedom

	50%	90%	95%	99%	99.9%
DF=5	0.73	2.02	2.57	4.03	6.87
DF=10	0.70	1.81	2.23	3.17	4.59
DF=20	0.69	1.72	2.09	2.85	3.85
DF=30	0.68	1.70	2.04	2.75	3.65
DF=50	0.68	1.68	2.01	2.68	3.50
(Normal)	0.67	1.64	1.96	2.58	3.29

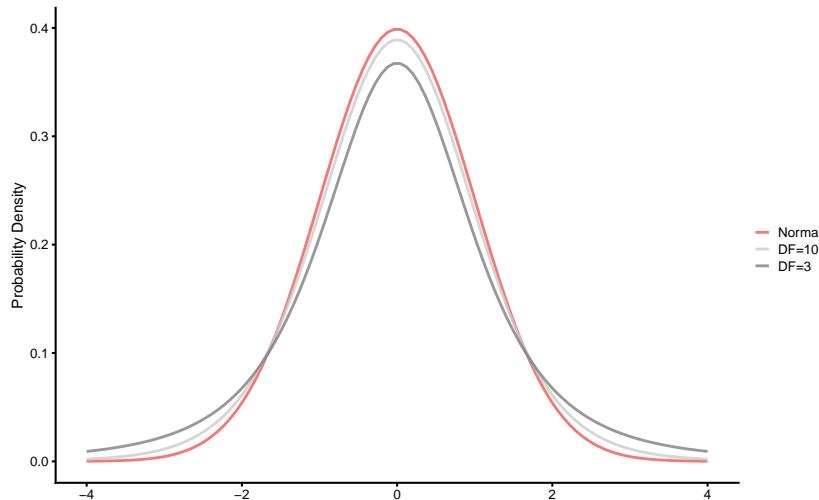


Figure 6.2: Student's t-distribution

Since Student's t distribution is fatter on the tails, critical values that cover 90, 95, and 99% of distribution mass are different than the commonly used ones for the normal distribution. Table 6.1 contains critical values for different DF. For example, 90% of the sampling distribution will be inside the $\bar{x} \pm 1.64 \times SEM$ interval for the normal distribution, but for Student t with DF=5, 90% of the sampling distribution will be inside the $\bar{x} \pm 2.02 \times SEM$ interval.

6.1 Null-Hypothesis Significance Testing

There are two approaches to statistical inference, be it frequentist or Bayesian: *hypothesis testing* and *estimation* (Cumming 2014; Kruschke and Liddell 2018b).

I will focus on the former, although latter will be covered as well. For the frequentist inference, mathematics behind both of these are the same and both involve standard errors.

Null-hypothesis significance testing (NHST) is still one of the most dominant approaches to statistical inference, although heavily criticized (for example see (Cumming 2014; Kruschke and Liddell 2018b)). In Figure 6.1, we have sampled from the known population, but in practice we don't know the true parameter values in the population, nor we are able to collect data from the whole population (unless it is a small one, but there is no need for statistical inference then, since the whole population is represented in our sample). Thus, we need to use sampled data to make inferences about the population. With NHST we want to *test* sample parameter or estimator (i.e. `mean` in this case) against the null-hypothesis (H_0). Null-hypothesis usually takes the *no effect* value, but it can take any value of interest for the researcher.

Although this sounds mouthful, a simple example will make it clearer. Imagine that we do know the true population `mean` height, but in one particular region the `mean` height of the sample differs from the known population `mean`. If we assume that this region belongs to the same population, then we want to test to see how likely are we to sample `mean` we have acquired or more extreme.

Figure 6.3 contains known population `mean` height as the null-hypothesis and estimated probabilities of observing sample `mean` of 180, 182.5, and 185cm (or $+2.2$, $+4.7$, $+7.2$ cm difference) *or larger* for sample sizes $N=5$, $N=10$ and $N=20$. Panel A on Figure 6.3 depicts *one-sided* approach for estimating probability of observing these sample `mean` heights. One-sided approach is used when we are certain about the direction of the effect. *Two-sided* approach, on the other hand, calculates probability for the effect of the unknown direction. In this example that would be sample `mean` height difference of ± 2.2 , ± 4.7 , ± 7.2 cm or larger. Two-sided approach is depicted on the Panel B (Figure 6.3).

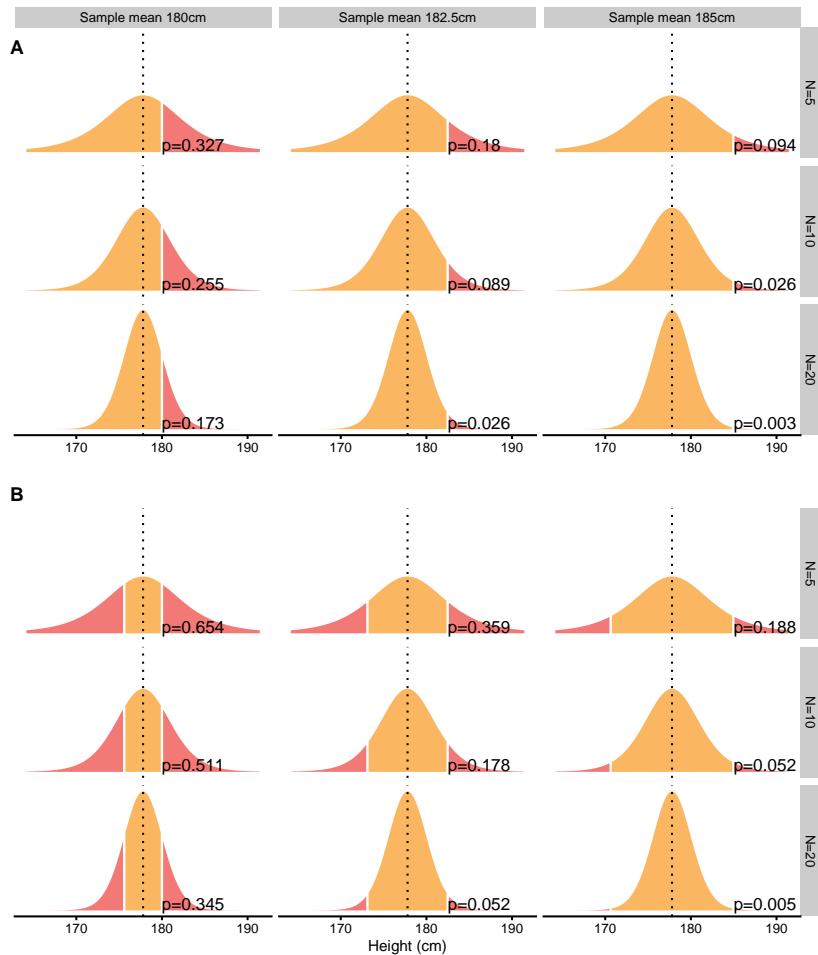


Figure 6.3: Null-hypothesis significance testing. Assuming null-hypothesis (true parameter value, or parameter value in the population, in this case `mean` or μ) is true, probability of observing sample parameter of a given magnitude or larger, is estimated by calculating proportion of sampling distribution that is over sample parameter value. The larger the sample size, the smaller the width of the sampling distribution. **A.** One-sided approach is used when we are certain about the direction of the effect. **B.** Two-sided approach is used when expected direction of the effect is unknown

The calculated probability of observing sample mean or larger, given null-hypothesis, is called *p-value*. In other words, p-value is the probability of observing data (in this case sample `mean`) given the null hypothesis (Equation (6.3))

$$p \text{ value} = p(\text{Data}|H_0) \quad (6.3)$$

It is easy to interpret p-values as “probability of the null hypothesis (given data)” ($p(H_0|\text{Data})$), but that is erroneous. This is Bayesian interpretation (also called *inverse probability*) which is quite common, even for the experienced researchers. Unfortunately, p-values cannot be interpreted in such way, but rather as a “probability of data given null hypothesis”.

As you can see from the Figure 6.3), for the same difference in sample **mean** height, different sample sizes will produce different p-values. This is because sampling distribution of the **mean** will be narrower (i.e. smaller **SEM**) as the sample size increase. In other words, for the same effect (in this case sample **mean**), p-value will be smaller as the sample size gets bigger. It is thus important to realize that p-values don't tell us anything about the magnitude of the effect (in this case the difference between the sample **mean** and the known population **mean**).

The procedures of acquiring p-values are called *statistical tests*. With the example above, we are using one variation of the *Student t test*, where we are calculating the test value t using the following Equation (6.4).

$$\begin{aligned} t &= \frac{\bar{x} - \mu}{\text{SEM}} \\ t &= \frac{\bar{x} - \mu}{\frac{SD}{\sqrt{N}}} \end{aligned} \quad (6.4)$$

P-value is then estimated by using the calculated t value and appropriate Student's t distribution (see Figure 6.2) to calculate the surface area over a particular value of t . This is usually done in the statistical program, or by using tables similar to Table 6.1 .

Once the p-value is estimated, we need to decide whether to reject the null-hypothesis or not. In order to do that, we need to define the error we are willing to accept. This error is called *alpha* (Greek α) or *Type I* error and refers to making an error of rejecting the null-hypothesis when null-hypothesis is true. Out of sheer convenience, alpha is set to 0.1 (10% error), 0.05 (5% error) or 0.01 (1% error).

If p-value is smaller than alpha, we will reject the null-hypothesis and state that the effect has *statistical significance*. The statistical significance has bad wording since it does not imply magnitude of the effect, only that the sample data come from a different population assumed by null-hypothesis.

Take the following example. Let's assume we have sample size of $N=20$ where sample **mean** is equal to 185cm. Using the known population **mean** (177.8cm) and

SD (10.16cm), we get that $t = 3.17$. Using two-sided test and alpha=0.05, can we reject the null-hypothesis? In order to do this we can refer to Table 6.1 and check that for DF=20 (which is not exact, but it will serve the purpose), 95% of sampling distribution (which leaves 2.5% on each tail which is equal to 5% alpha) will be within ± 2.08 . Since calculated $t = 3.17$ is over ± 2.08 , we can reject the null-hypothesis with alpha=0.05. Figure 6.3 (Panel B) depicts the exact p-value for this example, which is equal to $p=0.005$. Statistical software can calculate exact p-values, but before these were available, tables and procedure just describes were used instead.

6.2 Statistical Power

There is other type of error that we can commit: *Type II* error or *beta* (Greek β). In order to understand Type II error, we need to assume alternate hypothesis or H_a . Type II error refers to the error we make when we reject the alternate-hypothesis when alternate-hypothesis is true. Type I and Type II error are inversely related - the more we are willing to make Type I errors, the less likely we are going to make Type II errors, and *vice versa* (Table 6.2).

Table 6.2: **Type I and Type II errors**

	True H_0	True H_a
Rejected H_0	Type I	
Rejected H_a		Type II

It is important to keep in mind that with NHST, we never *accept* any hypothesis, we either reject it or not. For example, we never say “null-hypothesis is accepted ($p=0.23$)”, but rather “null-hypothesis is not rejected ($p=0.23$)”.

Assuming that alternate-hypothesis is true, probability of rejecting the null-hypothesis is equal to $1 - \beta$. This is called *statistical power* and depends on the magnitude of the effect we are aiming to detect (or not-to-reject to correct myself). Figure 6.4 depicts multiple examples of one-sided and two-sided statistical power calculations given the known alpha of 0.05 and null-hypothesis for difference in sample mean height of ± 2.5 , ± 5 , and ± 7.5 cm (+2.5, +5, and +7.5cm for one sided test) for N=5, N=10 and N=20.

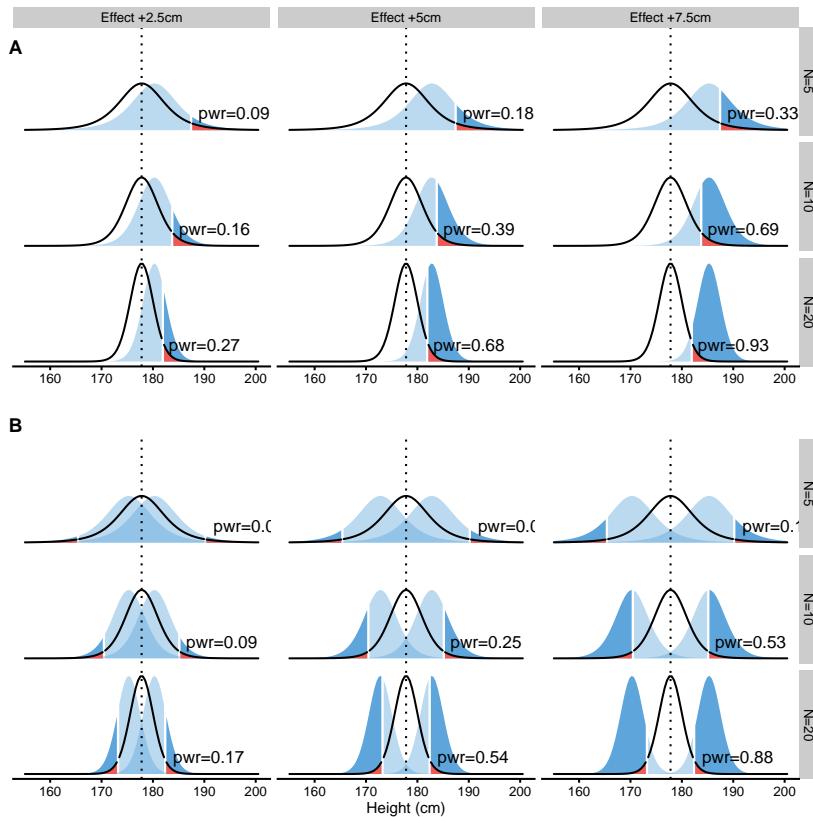


Figure 6.4: **Statistical power.** Statistical power is probability of detecting an effect of particular magnitude or larger. Visually, statistical power is dark blue surface and represents probability of rejecting the null-hypothesis given that the alternative hypothesis is true. **A.** One-sided approach. **B.** Two-sided approach

As can be seen from the Figure 6.4, the higher the magnitude of the effect (in this case difference in height means), the more likely we are to detect it (by rejecting null-hypothesis). Statistical power is mostly used when planning the studies to estimate sample size needed to detect effects of magnitude of interest (usually using known or observed effect from previous studies, or even SESOI). For example, question such as “How big of a sample do I need to detect 2.5cm difference with 80% power, alpha 0.05 and expected sample SD of 10cm?” is answered by using statistical power analysis. Statistical power, or sample size for needed statistical power can be easily calculated for simple analysis, but for some more elaborate analyses simulations are needed.

The frequentist approach to statistical inference is all about maintaining accepted error rates, particularly Type I, for both tests and estimates. This can be particularly difficult when *family-wise error rates* need to be controlled, and

these can emerge when multiple NHST are done. Some techniques, called p-hacking, can also introduce bias in the error rates by *fishing* for p-values (e.g. collecting samples until significant results are found). These topics are beyond the scope of this paper, but one of the reasons why some researchers prefer [Bayesian perspective](#).

6.3 New Statistics: Confidence Intervals and Estimation

Rather than performing NHST, uncertainty of the estimated parameter can be represented with the *confidence interval* (CI). CIs are usually pretty hard to explain and non-intuitive since they do not carry any distributional information.¹ It is thus better to refer to CIs as *compatibility intervals* (Gelman and Greenland 2019), since, let's say 95% confidence interval contains all the hypotheses parameter values that would not be rejected by $p < 0.05$ NHST (Kruschke and Liddell 2018b). This implies that, in the long-run (when sampling is repeated infinite number of times), 95% confidence interval will capture true parameter value 95% of the time.

Assuming $N=20$ samples come from the population where the true `mean` height is equal to 177.8cm and `SD` is equal to 10.16cm, calculated 95% CIs around sample parameter estimate (in this case sample `mean`), in the long run, will capture true population parameter 95% of the time. Figure 6.5 depicts first 100 samples out of total of 1,000 taken from the population with calculated 95% CIs. CIs that missed the true population parameter value are depicted in red. Table 6.3 contain the summary for this simulation. If this simulation is repeated for many more times, CIs will capture true population parameter 95% of the time, or in other words, have Type I error of 5%.

¹It is quite common to erroneously interpret CIs as Bayesian *credible intervals* (Kruschke and Liddell 2018b; McElreath 2015).

Table 6.3: Type I errors in 1000 samples

Sample	Correct %	Type I Errors %
1000	95.9	4.1

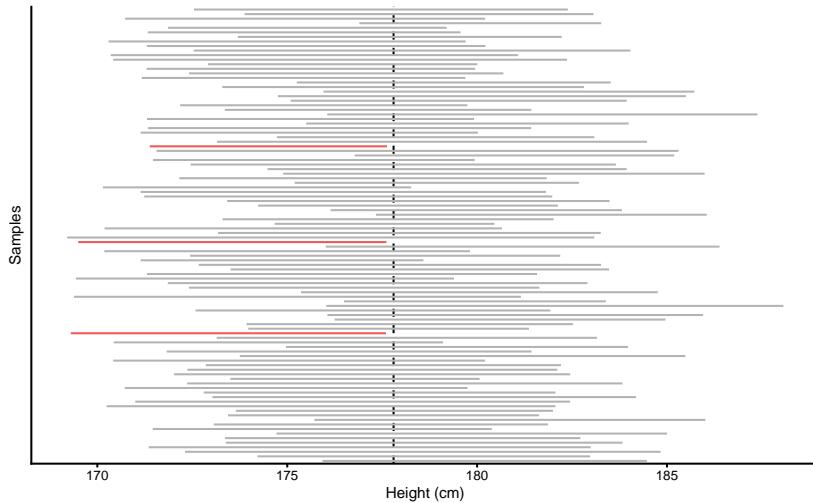


Figure 6.5: 95% confidence intervals for the sample mean, estimated for 100 samples (N=20 observations) drawn from population of known parameters (population mean is indicated by vertical line). In the long-run, 95% confidence intervals will span the true population value 95% of the time. Confidence intervals that didn't span the true population parameter value are colored in red

Similarly to different alphas, CIs can use different levels of confidence, usually 90%, 95%, 99%. As already mentioned, mathematics behind the confidence intervals is equal to mathematics behind NHST. In order to calculate two-sided CIs for the sample mean, the Equation (6.5) is used:

$$CI = \bar{x} \pm t_{crit} \times \widehat{SEM} \quad (6.5)$$

t_{crit} can be found in the Table 6.1, where for 95% two-sided confidence and DF=20, is equal to 2.086. Using the example of observed sample mean of 185cm, known sample SD (10.16cm) and N=20 (which is equal to DF=19, but for the sake of example DF=20 will be used), calculated 95% confidence interval is equal to 180.26 to 189.74cm. From the compatibility interpretation standpoint, this CI means that the hypotheses with values ranging from 180.26 to 189.74cm, will not be rejected with alpha=0.05.

Confidence intervals are great solution for visualizing uncertainties around estimates. Figure 6.6 depicts already used example in Figure 6.3 (two-sided and one-sided p-values), but this time 95% CIs around the sample means are depicted. Please note that in scenarios where 95% CIs cross the null-hypothesis, NHST will yield $p>0.05$. This means that null-hypothesis is not rejected and results are not statistically significant. CIs can be thus used to visually inspect and conclude whether or not the null-hypothesis would be rejected or not if NHST is performed.

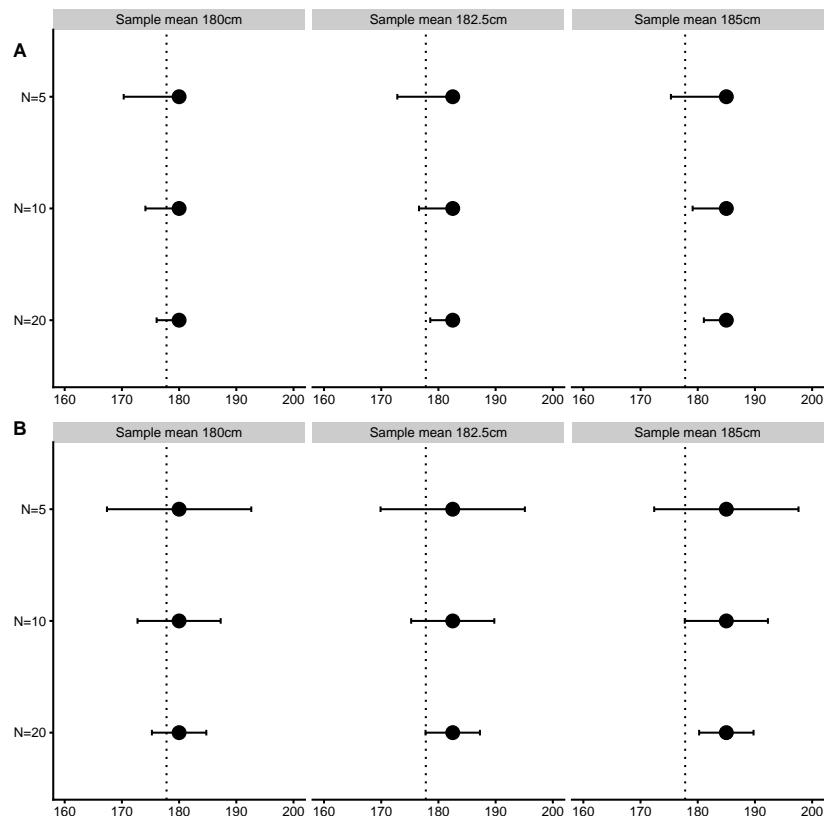


Figure 6.6: **95% Confidence intervals for sample mean.** Null-hypothesis of the population parameter value is indicated by vertical dashed line. If the 95% confidence interval doesn't touch or cross the null-hypothesis parameter value, p -value is less than 0.05 and effect is statistically significant (given alpha of 0.05). **A.** One-sided confidence intervals. **B.** Two-sided confidence intervals

6.4 Minimum Effect Tests

NHST doesn't tell us anything about the magnitudes of the effect. Just because the test is statistically significant ($p < 0.05$), it's doesn't imply practically meaningful effect. Rather than using null-hypothesis of *no effect*, we can perform numerous one-sided NHSTs by using SESOI thresholds to infer practical significance. These are called *minimum effect tests* (METs) and can distinguish between 6 different conclusions: *lower*, *not-higher*, *equivalent*, *not-lower*, *higher*, and *equivocal* effect. Figure 6.7 depicts how SESOI and CIs can be used to distinguish between these 6 magnitude-based conclusions (Barker and R. Schofield 2008; Sainani et al. 2019).

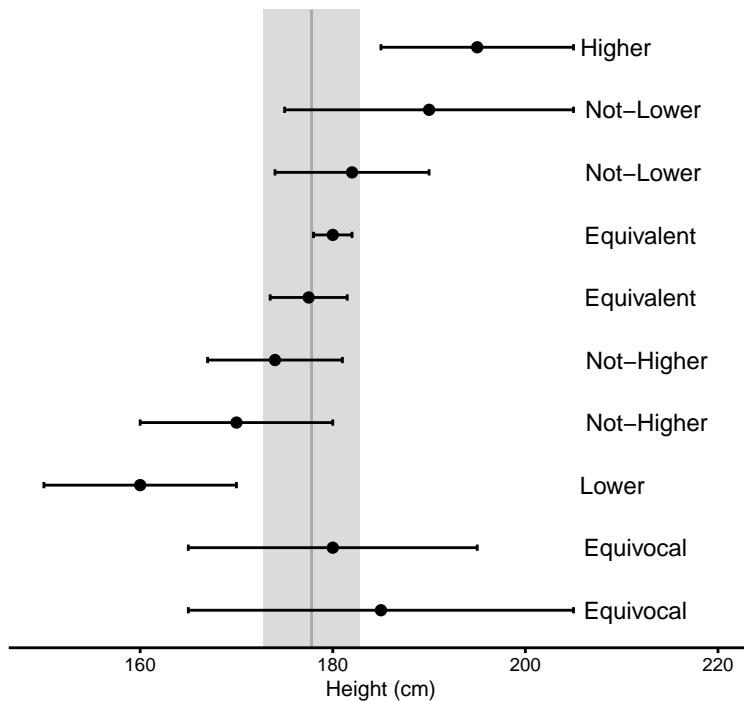


Figure 6.7: **Inference about magnitudes of effects.** Error bars represent confidence intervals around estimate of interest. Adapted and modified from Barker and R. Schofield (2008); Sainani et al. (2019)

6.4.1 Individual vs. Parameter SESOI

So far we have used SESOI to infer practically significant differences or changes at the *individual* level. For example, answering what is the practically meaningful

difference in height, SESOI was used to calculate proportions and chances of observing individuals with lower, equivalent and higher magnitudes of effects.

In prediction tasks, SESOI was used to infer practically meaningful prediction error. This helped answering the question regarding whether the individual predictions are within practically equivalent region.

However, apart from using SESOI to infer individual change, difference, and prediction magnitudes, SESOI can also be used to evaluate statistics or parameters against practically significant anchor. For example, in Equation (2.14), we have divided **mean** group difference with SESOI to create magnitude-based estimator. But here, we assumed that the same magnitude used at the individual level is of equal practical importance at the group level (i.e. aggregate level using the **mean** estimator). For example, individual change of $\pm 5\text{kg}$ might be practically important at the level of the individual, but not at the level of the group (i.e. parameter), and *vice versa*. Usually, they are assumed to be the same (see also [Ergodicity](#) section).

Since sample **mean** difference is the estimator of the parameter in the population we are interested in estimating, we tend to use SESOI to give practical anchors for parameters as well. It could be argued that different terms should be used for the parameter SESOI (particularly for standardized estimators such as [Cohen's d](#)) *versus* individual SESOI. For example, we can use ROPE term for parameters ([Kruschke and Liddell 2018a, 2018b](#)), and SESOI for individual-level magnitude inferences. For practical purposes they are considered equal, although I believe further discussion about this distinction is warranted, but outside the scope of this book.

6.4.2 Two one-sided tests of equivalence

Besides testing again null-hypothesis of no-effect, we can use the two one-sided tests (TOST) procedure to test for *equivalence* and reject the presence of the smallest effect size of interest (SESOI) ([Lakens, Scheel, and Isager 2018; Lakens 2017](#)). TOST involves using two one-sided NHSTs assuming parameter values at SESOI thresholds (Figure 6.8). Since the TOST produces two p-values, the larger of the two is reported. A conclusion of statistical equivalence is warranted when the larger of the two p-values is smaller than alpha ([Lakens 2017](#)).

From estimation perspective, statistical equivalence at the level of $\alpha=0.05$ can be inferred if the 90% (90% not 95%; it is not a typo) CI falls completely within SESOI band.

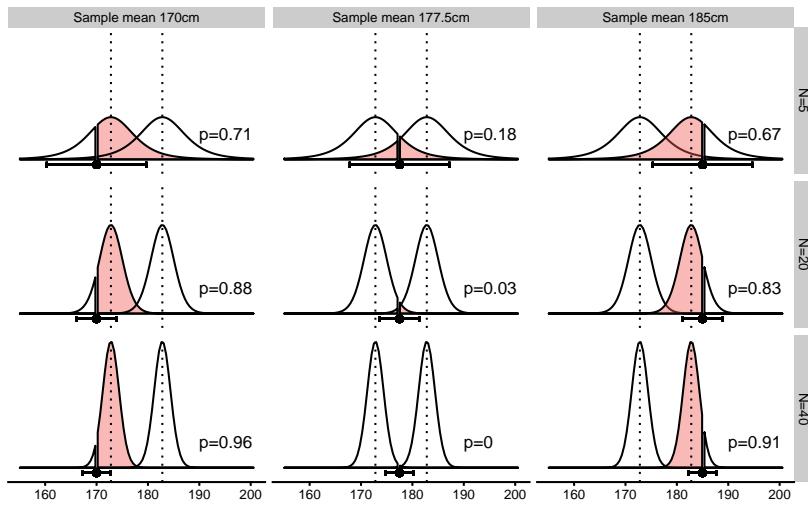


Figure 6.8: **Equivalence test using two one-sided tests (TOST).** Equivalence test involves two NHSTs at SESOI thresholds and calculates two one-sided p-values, out of which a larger one is reported as result. Error bars represent 90% confidence intervals.

6.4.3 Superiority and Non-Inferiority

Two same NHSTs at SESOI thresholds are utilized to test superiority and non-inferiority of the effects. In other words, we want to conclude whether the effect is higher and/or not-lower than SESOI. To achieve this, two one-sided NHSTs are performed to estimate the probability of observing effect in the positive direction (Figure 6.9).

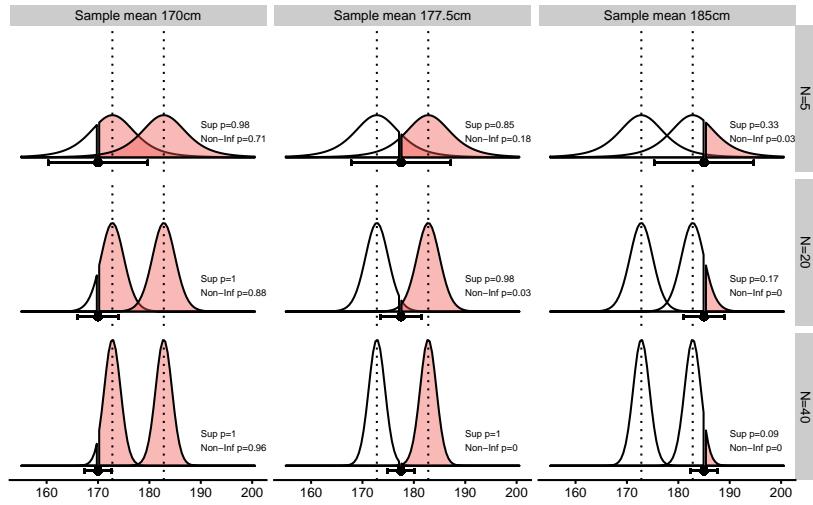


Figure 6.9: Superiority and Non-Inferiority tests. Similar to equivalence test using TOST procedure, superiority and non-inferiority tests involve two one-sided NHSTs at SESOI thresholds in the positive direction. Error bars represent 90% confidence intervals.

6.4.4 Inferiority and Non-Superiority

To test the inferiority and non-superiority of the effects, two one-sided NHSTs are performed to estimate the probability of observing effect in the negative direction (Figure 6.10).

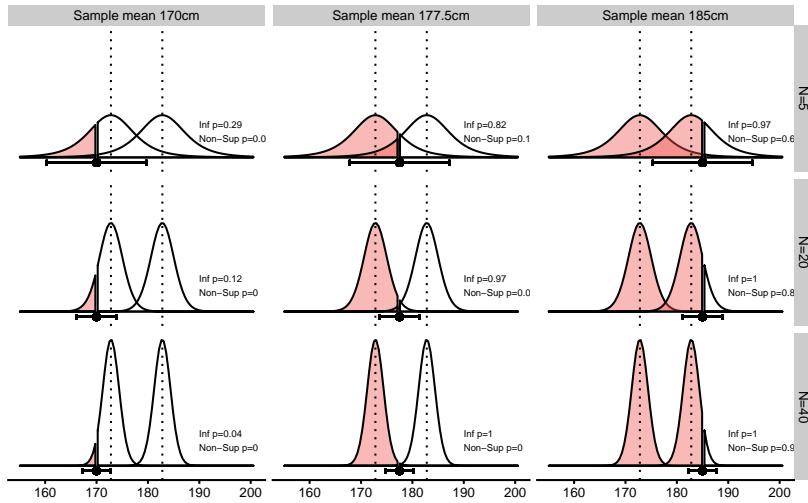


Figure 6.10: **Inferiority and Non-Superiority tests.** Similar to equivalence test using TOST procedure, inferiority and non-superiority tests involve two one-sided NHSTs at SESOI thresholds in the negative direction. Error bars represent 90% confidence intervals.

6.4.5 Inference from METs

The aforementioned METs provide five p-values: for lower (inferiority), not-higher (non-superiority), equivalent (equivalence), not-lower (non-inferiority), and higher (superiority) effect magnitude. These p-values can be used to make magnitude-based inferences about the effects. Figure 6.11 depicts already used examples to calculate p-values from METs and the final inference on the magnitude of the effect (see Figure 6.7).

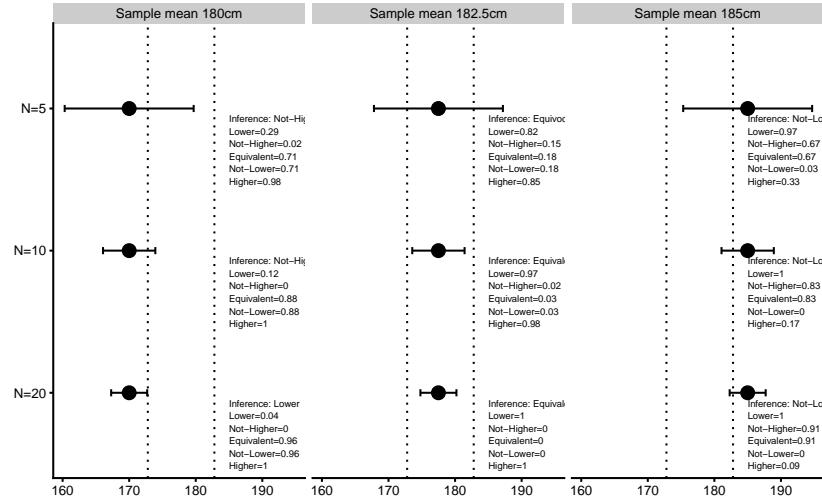


Figure 6.11: **Minimum Effect Test results.** Error bars represent 90% confidence intervals.

6.5 Magnitude Based Inference

Batterham and Hopkins (Batterham and Hopkins 2006; Hopkins et al. 2009) proposed novel approach in making meaningful inference about magnitudes, called *magnitude based inference* (MBI). MBI has been recently criticized (Barker and R. Schofield 2008; Borg et al. 2018; Curran-Everett 2018; Hopkins and Batterham 2018; Nevill et al. 2018; Sainani et al. 2019; Sainani 2018; Welsh and Knight 2015) for interpreting CIs as Bayesian credible intervals and for not controlling Type I and Type II errors.

As explained, CIs doesn't contain any probability distribution information about the true parameter. Although CIs, Bayesian *credible intervals* (with flat or non-informative *prior*), and *bootstrap CIs* tend to converge to the approximately same values for very simple tests (such as t-test for the sample mean), interpreting CIs established using frequentist approach as Bayesian credible intervals is not valid approach to statistical inference (Sainani et al. 2019). Figure 6.12 depicts Bayesian interpretation of the confidence intervals used in MBI.

Using MBI as a simple descriptive approach to interpret CIs can be rationalized, but making inferences from estimated probabilities is not recommended (Caldwell and Cheuvront 2019). If frequentist approaches are used for magnitude-based statistical inference, METs should be used instead.

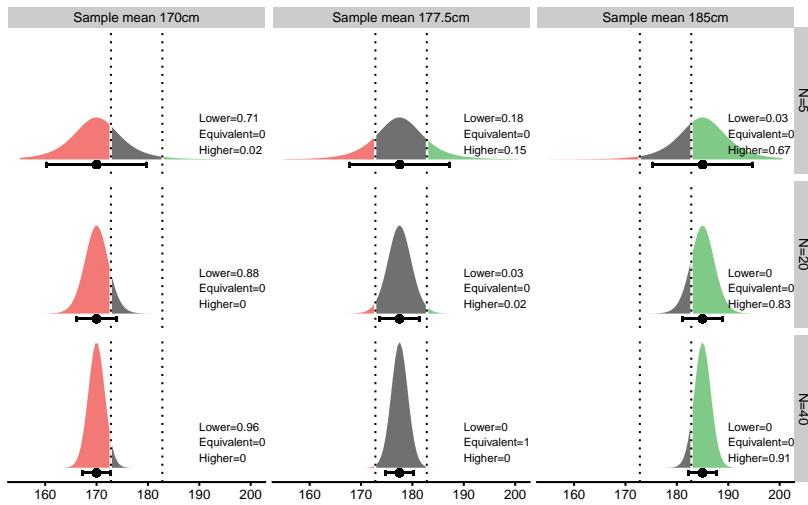


Figure 6.12: Magnitude-based inference use inappropriate Bayesian interpretation of the confidence intervals to calculate lower, equivalent, and higher probabilities of the effect. Vertical dashed lines represent SESOI thresholds. Error bars represent 90% confidence intervals.

There are numerous problems with frequentist inference (Kruschke and Liddell 2018b, 2018a). The results are not intuitive and are usually erroneously interpreted from the Bayesian perspective. Error rates need to be controlled for and adjusted when multiple comparisons are made, or when different *stopping* techniques are used, sampling distributions are unknown for some estimators and cannot be derived algebraically. Various assumptions such as assumptions of normality, non-colinearity and others, need to be made and tested for, and for more complex models, such as *hierarchical models*, p-values and CIs are only approximated (Kruschke and Liddell 2018a, 2018b). It is beyond this short chapter to delve into more details, regarding the frequentist approach to statistical inference, and readers are directed to references provided in this section.

Chapter 7

Bayesian perspective

Bayesian inference is reallocation of plausibility (credibility) across possibilities (Kruschke and Liddell 2018a, 2018b; McElreath 2015). Kruschke and Liddell (Kruschke and Liddell 2018a, 156) wrote in their paper as follows:

“The main idea of Bayesian analysis is simple and intuitive. There are some data to be explained, and we have a set of candidate explanations. Before knowing the new data, the candidate explanations have some prior credibility of being the best explanation. Then, when given the new data, we shift credibility toward the candidate explanations that better account for the data, and we shift credibility away from the candidate explanations that do not account well for the data. A mathematically compelling way to reallocate credibility is called Bayes’ rule. The rest is just details.”

The aim of this section is to provide the gross overview of the Bayesian inference using *grid approximation* method (McElreath 2015), which is excellent teaching tool, but very limited for Bayesian analysis beyond simple mean and simple linear regression inference. More elaborate discussion on the Bayesian methods, such as Bayes factor, priors selection, model comparison, and *Markov Chain Monte Carlo* sampling is beyond the scope of this book. Interested readers are directed to the references provided and suggested readings at the end of this book.

7.1 Grid approximation

To showcase the rationale behind Bayesian inference let’s consider the same example used in [Frequentist perspective](#) chapter - the male height. The question

Table 7.1: Parameter possibilities

mu	sigma
170	9
175	9
180	9
170	10
175	10
180	10
170	11
175	11
180	11

we are asking is, given our data, what is the true average male height (`mean`; mu or Greek letter μ) and `SD` (sigma or Greek letter σ). You can immediately notice the difference in the question asked. In the frequentist approach we are asking “What is the probability of observing the data¹ (estimator, like `mean` or Cohen's `d`), given the null hypothesis?”

True average male height and true `SD` represents parameters, and with Bayesian inference we want to relocate credibility across possibilities of those parameters (given the data collected). For the sake of this simplistic example, let's consider the following possibilities for the `mean` height: 170, 175, 180cm, and for `SD`: 9, 10, 11cm. This gives us the following *grid* (Table 7.1), which combines all possibilities in the parameters, hence the name grid approximation. Since we have three possibilities for each parameter, the grid consists of 9 total possibilities.

7.2 Priors

Before analyzing the collected data sample, with Bayesian inference we want to state the *prior* beliefs in parameter possibilities. For example, I might state that from previous data collected, I believe that the `mean` height is around 170 and 180cm, with a peak at 175cm (e.g. approximating normal curve). We will come back to topic of prior later on, but for now lets use *uniform* or *vague* prior, which assigns equal plausibility to all `mean` height and `SD` possibilities. Since each parameter has three possibilities, and since probabilities needs to sum up

¹Personally, I found this confusing when I started my journey into inferential statistics. I prefer to state “What is the probability of observing value (i.e. estimate) of the selected **estimator** given the null hypothesis (null being estimator value)?”, rather than using the term *data*. We are making inferences using the data estimator, not the data *per se*. For example, we are not inferring whether the groups differ (i.e. data), but whether the group `means` differ (estimator).

Table 7.2: Parameter possibilities with priors

mu	sigma	mu prior	sigma prior
170	9	0.33	0.33
175	9	0.33	0.33
180	9	0.33	0.33
170	10	0.33	0.33
175	10	0.33	0.33
180	10	0.33	0.33
170	11	0.33	0.33
175	11	0.33	0.33
180	11	0.33	0.33

to 1, each possibility has probability of 1/3 or 0.333. This is assigned to our grid in the Table 7.2.

7.3 Likelihood function

The sample height data we have collected for N=5 individuals is 167, 192, 183, 175, 177cm. From this sample we are interested in making inference to the true parameter values (i.e. `mean` and `SD`, or μ and σ). Without going into the *Bayes theorem* for *inverse probability*, the next major step is the *likelihood function*. Likelihood function gives us a likelihood of observing data, given parameters. Since we have 9 parameter possibilities, we are interested in calculating the likelihood of observing the data for each possibility. This is represented with a following Equation (7.1):

$$L(x|\mu, \sigma) = \prod_{i=1}^n f(x_i, \mu, \sigma) \quad (7.1)$$

The likelihood of observing the data is calculated by taking the *product* (indicated by $\prod_{i=1}^n$ sign in the Equation (7.1)) of likelihood of observing individual scores. The likelihood function is normal *probability density function* (PDF)²; whose parameters are μ and σ (see Figure 7.1). This function has the following Equation (7.2):

$$f(x_i, \mu, \sigma) = \frac{e^{-(x_i - \mu)^2 / (2\sigma^2)}}{\sigma \sqrt{2\pi}} \quad (7.2)$$

²There are other likelihood functions that one can use of course, similar to the various *loss functions* used in [Prediction](#) section.

Table 7.3: Likelihoods of observing scores given μ and σ equal to 175cm and 9cm

mu	sigma	x	likelihood
175	9	167	0.03
175	9	192	0.01
175	9	183	0.03
175	9	175	0.04
175	9	177	0.04

Table 7.4: Likelihoods and log likelihoods of observing scores given μ and σ equal to 175cm and 9cm

mu	sigma	x	likelihood	LL
175	9	167	0.03	-3.51
175	9	192	0.01	-4.90
175	9	183	0.03	-3.51
175	9	175	0.04	-3.12
175	9	177	0.04	-3.14

Let's take a particular possibility of μ and σ , e.g. 175cm and 9cm, and calculate likelihoods for each observed score (Table 7.3).

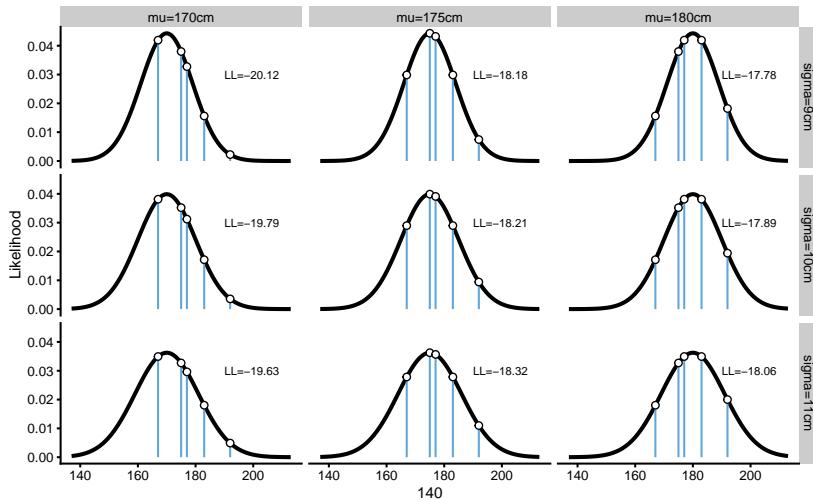
Now, to estimate likelihood of the sample, we need to take the product of each individual score likelihoods. However, now we have a problem, since the result will be very, very small number (1.272648×10^{-8}). To solve this issue, we take the log of the likelihood function. This is called *log likelihood* (LL) and it is easier to compute without the fear of losing digits. Table 7.4 contains calculated log from the score likelihood.

Rather than taking a product of the LL to calculate the overall likelihood of the sample, we take the sum. This is due the properties of the logarithmic algebra, where $\log x_1 \times x_2 = \log x_1 + \log x_2$, which means that if we take the exponent of the sum of the log likelihoods, we will get the same result as taking the exponent of the product of likelihoods. Thus the overall log likelihood of observing the sample is equal to -18.18. If we take the exponent of this, we will get the same results as the product of individual likelihoods, which is equal to 1.272648×10^{-8} . This *mathematical trick* is needed to prevent very small numbers and thus loosing precision.

If we repeat the same procedure for every parameter possibility in our grid, we get the following log likelihoods (Table 7.5). This procedure is also visually represented in the Figure 7.1 for easier comprehension.

Table 7.5: Sum of data log likelihoods for parameter possibilities

mu	sigma	mu prior	sigma prior	LL
170	9	0.33	0.33	-20.12
175	9	0.33	0.33	-18.18
180	9	0.33	0.33	-17.78
170	10	0.33	0.33	-19.79
175	10	0.33	0.33	-18.21
180	10	0.33	0.33	-17.89
170	11	0.33	0.33	-19.63
175	11	0.33	0.33	-18.32
180	11	0.33	0.33	-18.06

Figure 7.1: **Likelihood of data given parameters.** μ and σ represent parameters for which we want to estimate likelihood of observing data collected

7.4 Posterior probability

To get the *posterior* probabilities of parameter possibilities, likelihoods need to be multiplied with priors ($\text{posterior} = \text{prior} \times \text{likelihood}$). This is called *Bayesian updating*. Since we have log likelihoods, we need to sum the log likelihoods with log of priors instead ($\log \text{posterior} = \log \text{prior} + \log \text{likelihood}$). To get the posterior probability, after converting log posterior to posterior using exponent

Table 7.6: Estimated posterior probabilities for parameter possibilities given the data

mu	sigma	mu prior	sigma prior	LL	posterior
170	9	0.33	0.33	-20.12	0.02
175	9	0.33	0.33	-18.18	0.14
180	9	0.33	0.33	-17.78	0.20
170	10	0.33	0.33	-19.79	0.03
175	10	0.33	0.33	-18.21	0.13
180	10	0.33	0.33	-17.89	0.18
170	11	0.33	0.33	-19.63	0.03
175	11	0.33	0.33	-18.32	0.12
180	11	0.33	0.33	-18.06	0.15

Table 7.7: Joint distribution of the parameter possibilities. Sums at the table margins represent marginal probabilities

	170	175	180	Sum
9	0.02	0.14	0.20	0.36
10	0.03	0.13	0.18	0.34
11	0.03	0.12	0.15	0.30
Sum	0.08	0.38	0.54	1.00

($posterior = e^{\log posterior}$)³, we need to make sure that probabilities of parameter possibility sum to one. This is done by simply dividing probabilities for each parameter possibility by the sum of probabilities.

Table 7.6 contains the results of Bayesian inference. The posterior probabilities are called *joint probabilities* since they represent probability of a combination of particular μ and σ possibility.

Table 7.6 can be converted into 3x3 matrix, with possibilities of μ in the columns, and possibilities of the σ in the rows and posterior joint probabilities in the cells (Table 7.7). The *sums* of the joint probabilities in the Table 7.7 margins represent *marginal probabilities* for parameters.

Since we have only two parameters, the joint probabilities can be represented with the *heat map*. Figure 7.2 is a visual representation of the Table 7.7.

³There is one more *mathematical trick* done to avoid very small numbers explained in McElreath (2015) and it involves doing the following calculation to get the posterior probabilities: $posterior = e^{\log posterior - \max(\log posterior)}$.

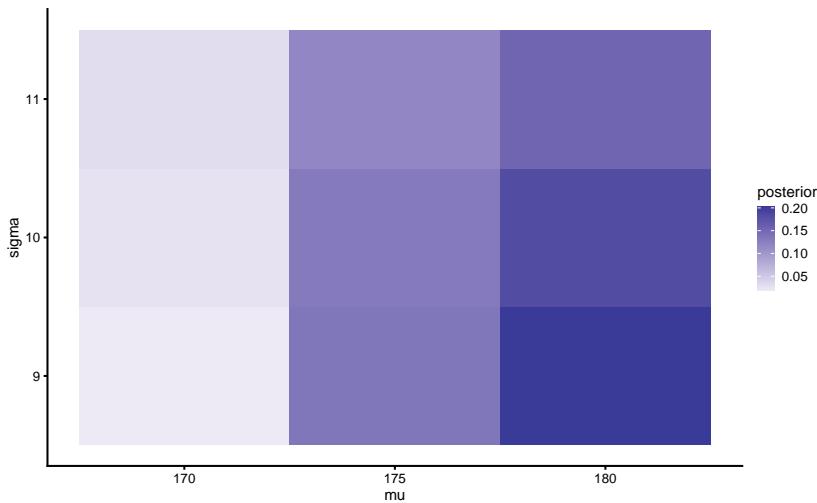


Figure 7.2: Heat map of μ and σ joint probabilities

When we have more than 2 parameters, visualization of joint probabilities get's tricky and we rely on visualizing marginal posterior probabilities of each parameter instead. As explained, marginal probabilities are calculated by summing all joint probabilities for a particular parameter possibility (see Table 7.7). Figure @ref(fig:(marginal-prior-posterior)) depicts marginal probabilities (including prior probabilities) for μ and σ .

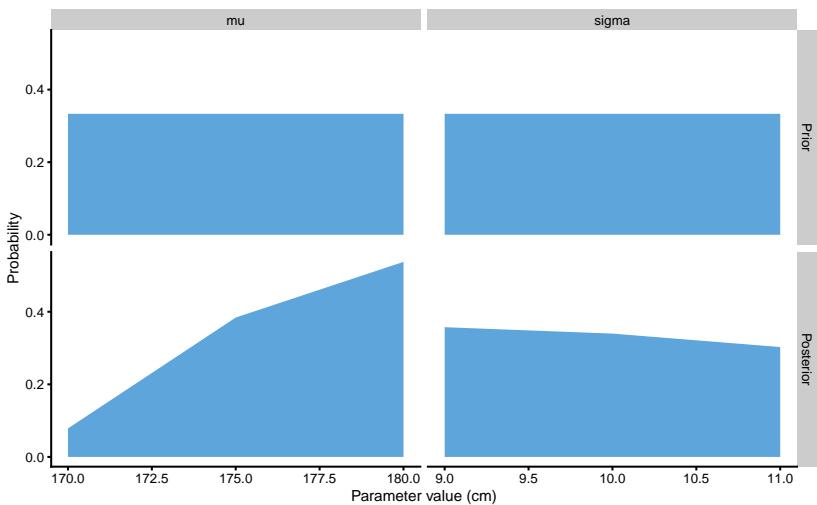


Figure 7.3: Prior and posterior distributions resulting from simplified grid-approximation example

As can be seen from the Figures 7.2 and 7.3, the most likely parameter possibilities, given the data, are μ of 180cm and σ of 9cm.

7.5 Adding more possibilities

So far, we have made this very granular in order to be understood. However, since we are dealing with continuous parameters, performing grid approximation for more than 9 total parameter possibilities seems warranted. The calculus is exactly the same, as well as the sample collected, but now we will use the larger range for both μ (160-200cm) and σ (1-30cm), each with 100 possibilities. We are estimating credibility for total of $100 \times 100 = 10,000$ parameter possibilities. Figure 7.4 depicts heat map for the joint probabilities, and Figure 7.5 depicts prior and posterior marginal distributions for each parameter.

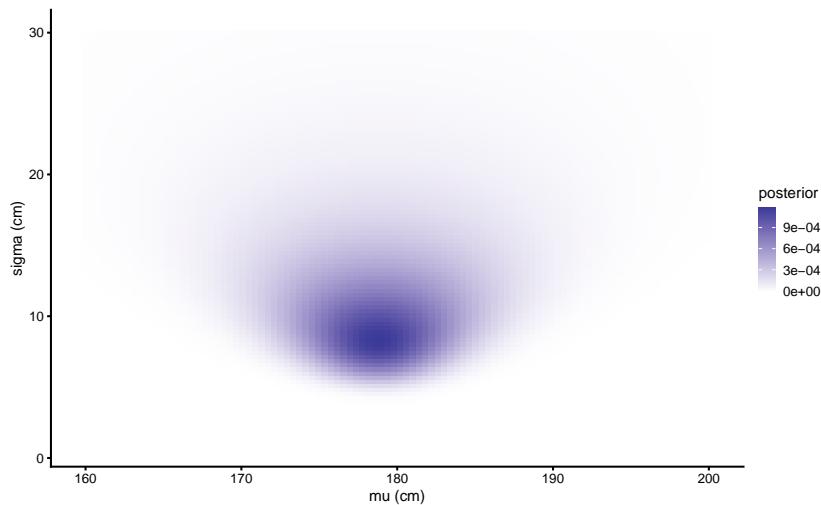


Figure 7.4: Heat map of μ and σ joint probabilities when 100×100 grid-approximation is used

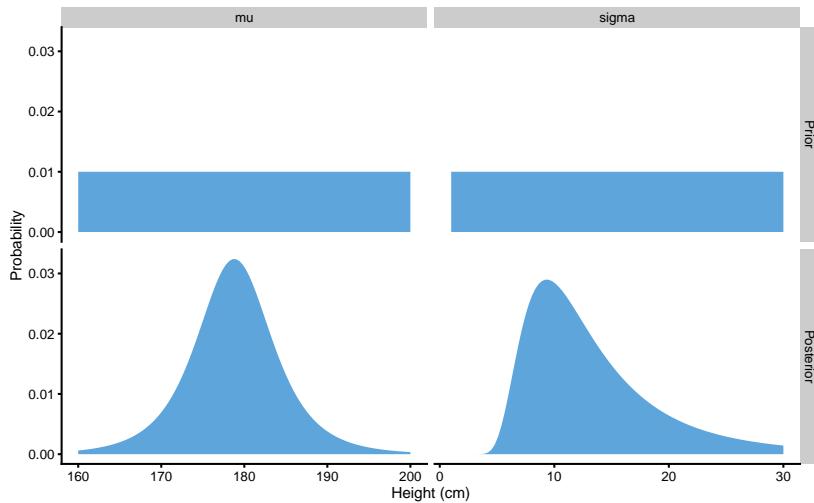


Figure 7.5: **Prior and posterior distributions resulting from 100×100 grid-approximation example**

Grid approximation utilized here is great for educational purposes and very simple models, but as number of parameters increases, the number of total parameter possibility grow so large, that it might take millions of years for a single computer to compute the posterior distributions. For example, if we have linear regression model with two predictors, we will have 4 parameters to estimate (intercept β_0 , predictor one β_1 , predictor two β_2 , and residual standard error σ), and if we use 100 possibilities for each parameter, we will get 10^8 total number of possibilities.

This was the reason why Bayesian inference was not very practical. Until algorithms such as *Markov Chain Monte Carlo* (MCMC) emerged making Bayesian inference a walk in the park. Statistical Rethinking book by Richard McElreath is outstanding introduction into these topics.

7.6 Different prior

In this example we have used vague priors for both μ and σ . But let's see what happens when I strongly believe (before seeing the data), that μ is around 190cm (using normal distribution with mean 190 and SD of 2 to represent this prior), but I do not have a clue about σ prior distribution and I choose to continue using uniform prior for this parameter.

This prior belief is, of course, wrong, but maybe I am biased since I originate, let's say from Montenegro, country with one of the tallest men. Figure 7.6 contains plotted prior and posterior distributions. As can be seen, using very

strong prior for μ shifted the posterior distribution to the higher heights. In other words, the data collected were not enough to *overcome* my prior belief about average height.

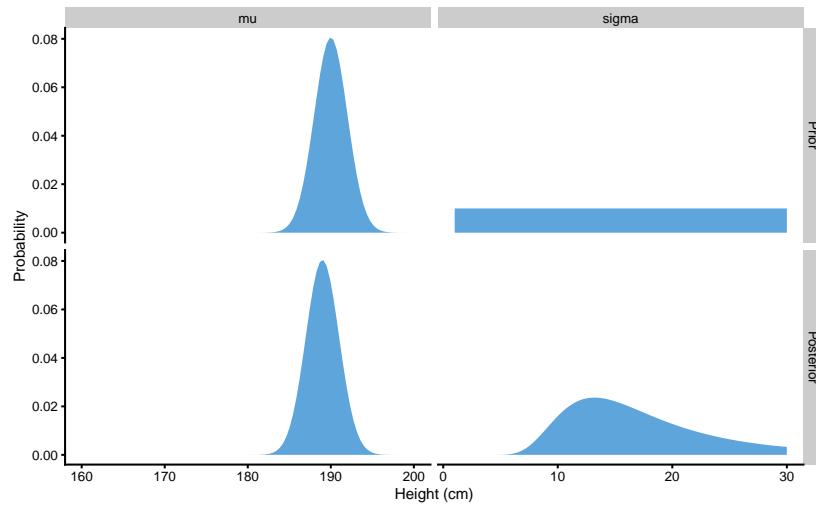


Figure 7.6: Effects of very strong prior on posterior

7.7 More data

The sample height data we have collected for $N=5$ individuals (167, 192, 183, 175, 177cm) was not strong to overcome prior belief. However, what if we sampled $N=100$ males from known population of known mean height of 177.8 and SD of 10.16? Figure 7.7 depicts prior and posterior distributions in this example. As can be seen, besides having narrower posterior distributions for μ and σ , more data was able to overcome my strong prior bias towards mean height of 190cm.

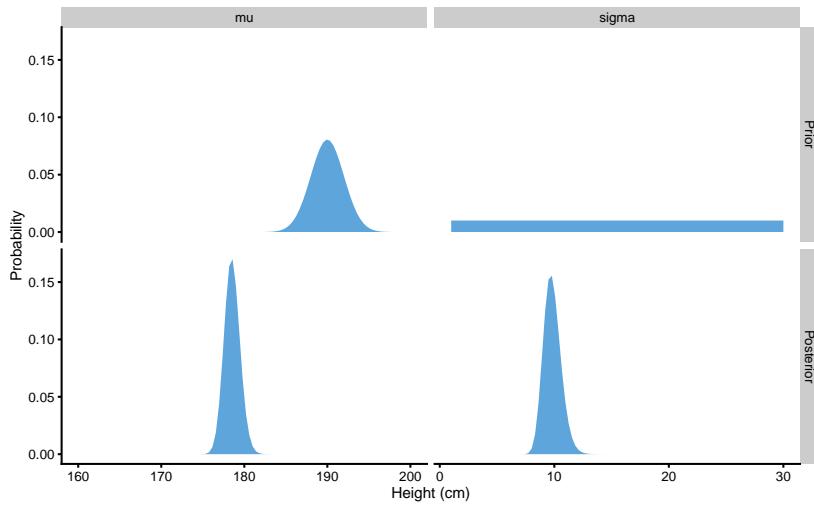


Figure 7.7: When larger sample is taken ($N=100$) as opposed to smaller sample ($N=20$), strong prior was not able to influence the posterior distribution

7.8 Summarizing prior and posterior distributions with MAP and HDI

In Bayesian statistics, the prior and posterior distributions are usually summarized using *highest maximum a posteriori* (MAP) and 90% or 95% *highest density interval* (HDI) (Kruschke and Liddell 2018a, 2018b; McElreath 2015). MAP is simply a `mode`, or the most probable point estimate. In other words, a point in the distribution with the highest probability. With normal distribution, MAP, `mean` and `median` are identical. The problems arise with distributions that are not symmetrical.

HDI is similar to frequentist CI, but represents an interval which contains all points within the interval that have higher probability density than points outside the interval (Makowski, Ben-Shachar, and Lüdecke 2019a, 2019b). HDI is more computationally expensive to estimate, but compared to *equal-tailed interval* (ETI) or *percentile interval*, that typically excludes 2.5% or 5% from each tail of the distribution (for 95% or 90% confidence respectively), HDI is not equal-tailed and therefore always includes the mode(s) of posterior distributions (Makowski, Ben-Shachar, and Lüdecke 2019a, 2019b).

Figure 7.8 depicts comparison between MAP and 90% HDI, `median` and 90% percentile interval or ETI, and `mean` and $\pm 1.64 \times SD$ for 90% confidence interval. As can be seen from the Figure 7.8, the distribution summaries differ since the distribution is asymmetrical and not-normal. Thus, in order to summarize prior

or posterior distribution, MAP and HDI are most often used, apart from visual representation.

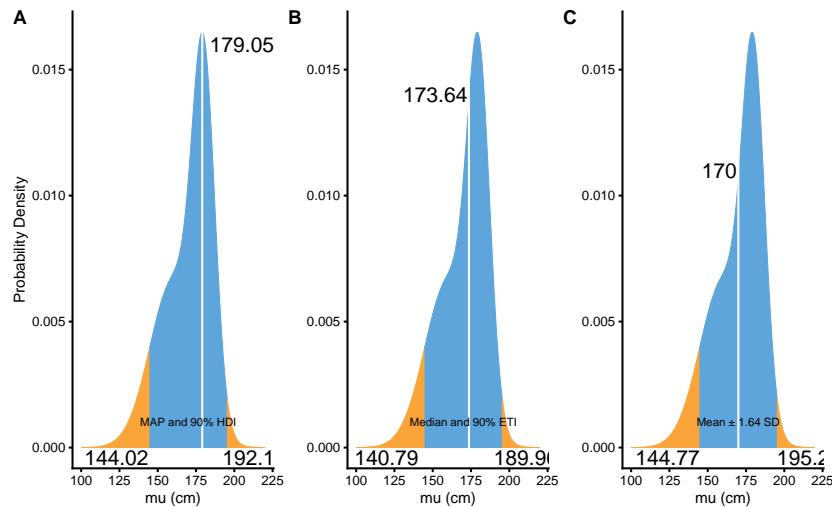


Figure 7.8: Summarizing prior and posterior distribution. A. MAP and 90% HDI. B. Median and 90% ETI. C. Mean and $\pm 1.64 \times SD$

Using SESOI as a trivial range, or as a ROPE (Kruschke and Liddell 2018a, 2018b), Bayesian equivalence test can be performed by quantifying proportion of posterior distribution inside the SESOI band (Kruschke and Liddell 2018a, 2018b; Makowski, Ben-Shachar, and Lüdecke 2019a; Makowski et al., n.d.). **Magnitude Based Inference** discussed in the the previous chapter, would also be valid way of describing the posterior distribution.

Besides estimating using MAP and HDI, Bayesian analysis also allows hypothesis testing using *Bayes factor* or *MAP based p-value* (Kruschke and Liddell 2018a, 2018b; Makowski, Ben-Shachar, and Lüdecke 2019a; Makowski et al., n.d.). Discussing these concepts is out of the range of this book and interested readers can refer to references provided for more information.

7.9 Comparison to NHST Type I errors

How do the Bayesian HDIs compare to frequentist CIs? What are the Type I error rates when the data is sampled from the null-hypothesis? To explore this question, I will repeat the simulation from [New Statistics: Confidence Intervals and Estimation](#) section, where 1,000 samples of N=20 observations are sampled from a population where the true mean height is equal to 177.8cm and SD is equal to 10.16cm. Type I error is committed when the the 95% CIs or 95% HDI

Table 7.8: Frequentist vs. Bayesian Type I errors

method	Sample	Correct %	Type I Errors %
Bayesian	1000	96.1	3.9
Frequentist	1000	95.5	4.5

intervals of the sample mean don't cross the true value in the population. Table 7.8 contains Type I errors for frequentist and Bayesian estimation.

As can be seen from the Table 7.8, frequentist CI and Bayesian HDI Type I error rate are not identical (which could be due to the grid approximation method as well as due to only 1000 samples used). This is often a concern, since Bayesian methods do not control error rates (Kruschke and Liddell 2018a). Although frequentist methods revolve around limiting the probability of Type I errors, error rates are extremely difficult to pin down, particularly for complex models, and because they are based on sampling and testing intentions (Kruschke and Liddell 2018a). For more detailed discussion and comparison of Bayesian and frequentist methods regarding the error control see Kruschke (2013); Wagenmakers (2007); Morey et al. (2016). Papers by Kristin Sainani (Sainani et al. 2019; Sainani 2018) are also worth pondering about which will help in understanding estimation and comparison of Type I and Type II error rates between different inferential methods, particularly when magnitude-based inference using SESOI is considered.

Chapter 8

Bootstrap

As already stated, some estimators have unknown sampling distribution, particularly those that might have more practical use and answer predictive questions by the practitioners (e.g. “what is the proportion of athletes I can expect to demonstrate beneficial response due to this treatment?”). Hence, the frequentist approach is very hard to use. With the Bayesian approach, some estimators might be really hard to be modeled and represented, researchers might be confused with the priors and likelihood function selections, there is knowledge needed to understand and diagnose sampling chains from MCMC algorithms and so forth.

Bootstrap comes for the rescue (Canty and Ripley 2017; A. C. Davison and Hinkley 1997b; Efron and Hastie 2016; Hesterberg 2015; Guillaume A Rousselet, Pernet, and Wilcox 2019b, 2019a). Bootstrap is very simple and intuitive technique that is very easy to carry out. Let’s take an example to demonstrate simplicity of the bootstrap. Continuing with the height example, let’s assume that the following sample is collected for N=10 individuals: 167, 175, 175, 176, 177, 181, 188, 190, 197, 211cm. We are interested in estimating the true `mean` in the population, `SD` in the population, and proportion of individuals taller than 185cm (`prop185`; using algebraic method and estimated `SD`). The first step, as explained in the [Description](#) section of this book, is to estimate those parameters using sample. But how do we estimate the uncertainty around the sample estimates?

Bootstrap involves *resampling* from the sample itself and then recalculating estimates of interest. If we have N=10 observations in the collected sample, for each bootstrap resample we are going to draw 10x1 observations. Some observations might be drawn multiple times, while some might not be drawn at all. This is then repeated numerous times, e.g., 2,000-10,000 times and for each bootstrap resample the estimators of interest are estimated. Table 8.1 contains 10 bootstrap resamples with calculated estimators of interest. Bootstrap resample of number 0 represents the original sample.

Table 8.1: Bootstrap resamples

Boot resample	Observations	mean	SD	prop185
0	167 175 175 176 177 181 188 190 197 211	183.7	13.00	0.46
1	167 167 167 177 188 188 188 197 197	182.4	11.98	0.41
2	175 175 175 181 181 181 190 197 197	183.3	8.49	0.42
3	175 175 177 177 181 181 188 188 190	182.0	5.98	0.31
4	167 181 181 181 188 188 197 197 197	185.8	9.61	0.53
5	167 176 176 177 181 188 190 190 211 211	186.7	14.71	0.55
6	167 167 167 175 177 177 188 188 190 211	180.7	13.88	0.38
7	175 175 175 176 177 177 181 181 188 190	179.5	5.50	0.16
8	175 175 175 176 177 181 188 190 190 190	181.7	6.96	0.32
9	176 176 177 181 190 190 190 190 197 211	187.8	10.97	0.60
10	167 167 175 175 175 176 176 177 177 188	175.3	5.83	0.05
11	175 177 181 181 181 188 190 197 197 211	187.8	11.21	0.60
12	175 175 175 176 176 177 181 188 190 211	182.4	11.47	0.41
13	175 175 177 181 181 188 190 190 211 211	187.9	13.43	0.59
14	167 175 175 176 177 177 188 190 211 211	184.7	15.34	0.49
15	175 177 177 181 181 188 188 190 211 211	187.9	13.21	0.59
16	167 175 175 175 177 177 181 188 190 197	180.2	8.92	0.30
17	175 175 175 176 177 177 181 190 190 197	181.3	8.04	0.32
18	167 175 175 175 176 177 177 188 188 188	178.6	7.07	0.18
19	167 167 175 177 181 188 190 197 197 211	185.0	14.24	0.50
20	167 167 175 175 176 176 177 181 197 211	180.2	13.66	0.36

If we repeat this procedure 10,000 times, we can visualize bootstrap distribution of the estimators (Figure 8.1).

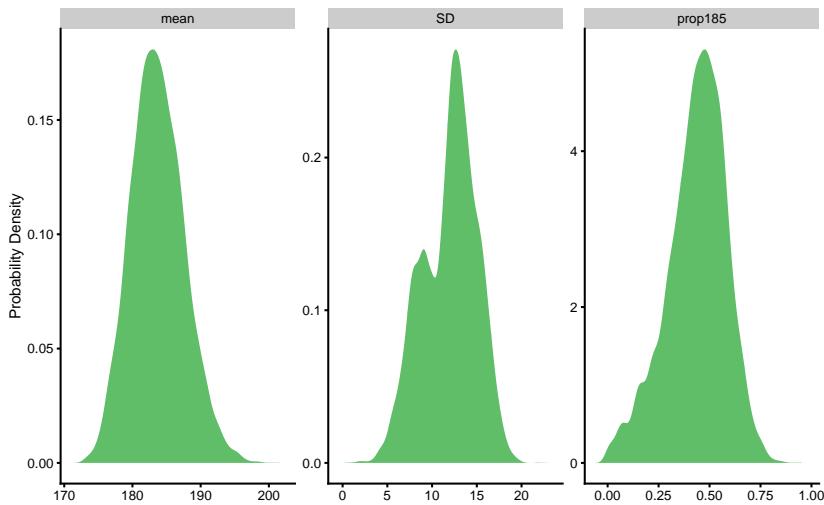


Figure 8.1: **Bootstrap distribution of the estimators using 10,000 resamples**

How should this bootstrap distribution be interpreted? In “Elements of Statistical Learning”, the following quote regarding bootstrap distribution can be found (Hastie, Tibshirani, and Friedman 2009, 272):

“In this sense, the bootstrap distribution represents an (approximate) nonparametric, noninformative posterior distribution for our parameter. But this bootstrap distribution is obtained painlessly — without having to formally specify a prior and without having to sample from the posterior distribution. Hence we might think of the bootstrap distribution as a “poor man’s” Bayes posterior. By perturbing the data, the bootstrap approximates the Bayesian effect of perturbing the parameters, and is typically much simpler to carry out”

Although the bootstrap was originally developed as a purely frequentist device (Efron 2005), as per the quote above, it can be treated as “poor man’s” Bayes posterior.

8.1 Summarizing bootstrap distribution

Bootstrap allows for both estimation and hypothesis testing. When it comes to estimations, point estimate of the bootstrap distribution is the sample parameter

estimate. Confidence intervals around sample estimate are usually calculated using percentile approach (or ETI), or other approaches such as *adjusted bootstrap percentile* (BCa) (Canty and Ripley 2017; A. C. Davison and Hinkley 1997b; Efron and Hastie 2016; Hesterberg 2015; Guillaume A Rousselet, Pernet, and Wilcox 2019b, 2019a), or even HDI as used with Bayesian posterior distributions. In this book I will utilize BCa intervals unless otherwise stated.

Hypothesis testing using the bootstrap distribution is possible through calculated p-value (Guillaume A Rousselet, Pernet, and Wilcox 2019b, 2019a). This not only allows for bootstrap NHST, but also all other MET, as well as MBI estimates (which assumes Bayesian interpretation of the bootstrap distributions). This is simply done by counting bootstrap sample estimates that are below or above particular threshold (i.e. null-hypothesis or SESOI). The R code (R Core Team 2018; RStudio Team 2016) below demonstrates how two-way NHST can be performed as well as probability of lower, equivalent, and higher effect given the SESOI thresholds.

```
null_hypothesis <- 0 # Value for the null

SESOI_lower <- -1 # threshold for the 'lower' effect magnitude
SESOI_upper <- 1 # threshold for the 'upper' effect magnitude

# Calculation of the p-value where boot.estimator is the
# bootstrap resample values for the estimator of interest
p_value <- mean(boot.estimator > null_hypothesis)
p_value <- p_value + 0.5 * mean(boot.estimator == null_hypothesis)
p_value <- 2 * min(c(p_value, 1 - p_value)) # Two-way p-value

# Calculate probability of lower, equivalent and higher
# effect magnitude
lower <- mean(boot.estimator < SESOI_lower)
higher <- mean(boot.estimator > SESOI_upper)
equivalent <- 1 - (lower + higher)
```

8.2 Bootstrap Type I errors

As we already did with the frequentist and Bayesian inference, let's get estimates of Type I errors for bootstrap method (10,000 bootstrap resamples) by drawing 1,000 samples of N=20 observations from the population where the true `mean` height is equal to 177.8cm and `SD` is equal to 10.16cm. Besides estimating Type I error for the sample `mean`, we can also estimate Type I errors for sample `SD` and `prop185`, since the true population values are known. In the case of `prop185`, the true population value is equal to 0.24. Type I error is committed when the 95% bootstrap CIs of the sample estimate don't cross the true value in the

Table 8.2: **Bootstrap Type I errors**

parameter	Sample	Correct %	Type I Errors %
mean	1000	92.7	7.3
sd	1000	89.9	10.1
prop185	1000	93.1	6.9

population. Figure 8.2 depicts the first 100 samples out of the total of 1,000, taken from the population with calculated 95% bootstrap CIs. CIs that missed the true population parameter value are depicted in red. Table 8.2 contains the summary for this simulation.

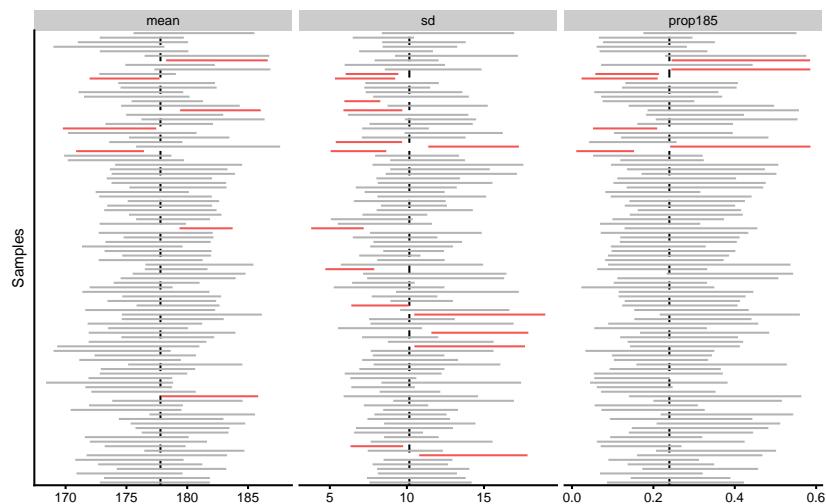


Figure 8.2: **Bootstrap 95%confidence intervals.** Intervals not capturing the true population parameter are colored in red

As can be seen from the Table 8.2, Type I error for the σ parameter is larger than expected. This could be due to the non-symmetrical bootstrap distribution that might not be perfectly represented with the BCa approach of calculating CIs.

I am not hiding my preference for the bootstrap methods due to their intuitive nature and simple usage for generating inferences for any estimator.

However, bootstrap is not panacea and there are caveats especially for the small samples sizes (Guillaume A Rousselet, Pernet, and Wilcox 2019b, 2019a; Wilcox, Peterson, and McNitt-Gray 2018; Wilcox and Rousselet 2017).

Chapter 9

Statistical inference conclusion

Which one is better, frequentist, Bayesian or bootstrap? Unfortunately, there is no objective, automatic way to statistical inference. All approaches demand careful planning and consideration of the research question of interest. All approaches rely on assumptions, some more clearly stated as with Bayesian approach, and some more inherent as with frequentist approach. All approaches to statistical inference represent Small Worlds that are hoped to help explain the Large World. Rather than approaching inferential statistics as something that represents and estimates *true state of the World*, inferential statistics might be approached as descriptive statistic (Amrhein, Trafimow, and Greenland 2019).

This section on statistical inference is best summarized with the following quotes:

“Statistical inference often fails to replicate. One reason is that many results may be selected for drawing inference because some threshold of a statistic like the P-value was crossed, leading to biased reported effect sizes. Nonetheless, considerable non-replication is to be expected even without selective reporting, and generalizations from single studies are rarely if ever warranted. Honestly reported results must vary from replication to replication because of varying assumption violations and random variation; excessive agreement itself would suggest deeper problems, such as failure to publish results in conflict with group expectations or desires. A general perception of a “replication crisis” may thus reflect failure to recognize that statistical tests not only test hypotheses, but countless assumptions and the entire environment in which research takes place. Because of all the uncertain and unknown assumptions that underpin statistical inferences, we should treat inferential statistics as highly unstable

local descriptions of relations between assumptions and data, rather than as providing generalizable inferences about hypotheses or models. And that means we should treat statistical results as being much more incomplete and uncertain than is currently the norm. Acknowledging this uncertainty could help reduce the allure of selective reporting: Since a small P-value could be large in a replication study, and a large P-value could be small, there is simply no need to selectively report studies based on statistical results. Rather than focusing our study reports on uncertain conclusions, we should thus focus on describing accurately how the study was conducted, what problems occurred, what data were obtained, what analysis methods were used and why, and what output those methods produced." (Amrhein, Trafimow, and Greenland 2019, Abstract, pp. 262)

"Exercising awareness of multiple perspectives, we emphasize that we do not believe that one of these philosophies is the correct or best one, nor do we claim that reducing the different approaches to a single one would be desirable. What is lacking here is not unification, but rather, often, transparency about which interpretation of probabilistic outcomes is intended when applying statistical modeling to specific problems. Particularly, we think that, depending on the situation, both "aleatory" or "epistemic" approaches to modeling uncertainty are legitimate and worthwhile, referring to data generating processes in observer-independent reality on one hand and rational degrees of belief on the other." (Gelman and Hennig 2017, 20)

"Broadly speaking, 19th century statistics was Bayesian while the 20th century was frequentist, at least from the point of view of most scientific practitioners. Here in the 21st century scientists are bringing statisticians much bigger problems to solve, often comprising millions of data points and thousands of parameters. Which statistical philosophy will dominate practice? My guess, backed up with some recent examples, is that a combination of Bayesian and frequentist ideas will be needed to deal with our increasingly intense scientific environment. This will be a challenging period for statisticians, both applied and theoretical, but it also opens the opportunity for a new golden age, rivaling that of Fisher, Neyman, and the other giants of the early 1900's." (Efron 2005)

This being said, my personal opinion is that more and more sport scientists will consider Bayesian analysis, particularly with the new tools that provide ease of Bayesian model definition and training, as well as visualization of the results. For the benefits of Bayesian over frequentist analysis please see the papers by Kruschke *et al.* (Kruschke and Liddell 2018a, 2018b) as well as suggested reading on Bayesian analysis at the end of this book.

Chapter 10

Measurement Error

Measurement error is involved in all measurements and causes an *observed score* to be different from the *true score* (Allen and Yen 2001; Novick 1966; Swinton et al. 2018; Borsboom 2009). This results in *measurement bias* affecting descriptive analysis, causal inferences (Hernán 2017; Hernán and Robins, n.d.; Hernan and Cole 2009), and predictive performances (Kuhn and Johnson 2018).

In mathematical notation, observed score (OS) comprises of the hypothetical true score (TS) and measurement error (ME) (Equation (10.1)) (Allen and Yen 2001; Swinton et al. 2018). This conceptual model is called *Classical Test Theory* (for more information, philosophy, and issues behind it please see Borsboom (2009))

$$OS = TS + ME \tag{10.1}$$

In the sports science domain, since the measured objects are usually humans, measurement error comprises of *instrumentation* and *biological noise* (Swinton et al. 2018). In this book I assume instrumentation noise to be error caused solely by the measurement apparatus (Swinton et al. 2018). Biological noise, on the other hand, is defined as an error in the observed scores caused by biological processes, including, but not limited to, phenomena such as circadian rhythm, nutritional intake, sleep and motivation (Swinton et al. 2018). Thus, I prefer to use the terms *instrumentation error* and *biological variation* (these will be further discussed in the Extending the Classical Test Theory section)

Both instrumentation and biological noises consist of two types of errors: *systematic error* and *random error* (Figure 10.1)¹.

¹These two types of error are conceptually equivalent to the bias and variance (see Prediction section), and systematic and stochastic intervention effects (see Causal inference section)

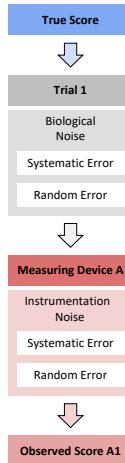


Figure 10.1: Measurement error components

Systematic error represents constant and stable error that is fixed across measurements. Systematic error is commonly referred to as *bias*. With measurement instruments that have a linear response, systematic error can be further divided into *proportional bias* and *fixed bias* (Will G Hopkins 2004b; Hopkins 2010, 2007). Random error (ϵ) represents unknown and unpredictable error, which varies between measurements. Random errors are often represented and modeled using a Gaussian normal distribution (with mean zero and SD which represent a parameter of the random error). The Equation (10.2) represent theoretical linear relationship between TS and OS, with normally distributed random error.

$$OS = Fixed\ Bias + (Proportional\ Bias \times TS) + \epsilon \quad (10.2)$$

This can be easily explained with a simple example. Imagine N=20 athletes being measured on a novel bodyweight scale, using total of 5 trials separated by 1 minute. The assumption in this example is that there is no change in the TS across trials (e.g. athletes are not allowed to use bathroom, consume water or food, nor change the wardrobe) and that there is no biological noise involved (i.e. there are no fluctuations in bodyweight due motivation, fatigue or what have you). Since this is simulated example, we know the TS of the athletes, but also the instrumentation noise characteristic of the novel bodyweight scale (i.e. this represents the underlying DGP). This scale tends to have proportional bias equal to factor 1.01 (i.e. athlete weighting 100kg which is his TS, will have OS equal to 101kg due proportional bias, while the athlete weighting 50kg will have her OS equal to 50.5kg), fixed bias equal to 1kg (everyone will get OS higher for 1kg than TS), and random error normally distributed with SD equal to 0.5kg. Equation (10.3) captures this relationship between OS and TS.

Table 10.1: **Simulated 5 trials from known true score and measurement error**

Athlete	TS (kg)	OS 1 (kg)	OS 2 (kg)	OS 3 (kg)	OS 4 (kg)	OS 5 (kg)
Athlete 01	77.93	79.47	79.37	79.25	80.38	79.46
Athlete 02	76.11	77.83	77.71	77.36	78.32	76.78
Athlete 03	77.04	78.65	78.30	78.48	78.30	78.55
Athlete 04	54.96	56.58	56.36	56.05	56.20	56.45
Athlete 05	84.03	85.53	85.44	86.10	85.96	85.11
Athlete 06	61.32	61.84	63.05	63.52	63.73	62.43
Athlete 07	68.62	70.26	69.82	70.34	70.45	70.84
Athlete 08	61.06	62.03	62.91	62.93	62.15	62.40
Athlete 09	80.46	81.34	83.04	82.57	82.60	82.33
Athlete 10	91.14	94.02	93.47	93.28	93.40	93.51
Athlete 11	79.98	81.89	82.13	81.13	81.57	81.94
Athlete 12	67.07	69.54	68.32	68.29	69.23	67.72
Athlete 13	79.41	80.66	80.81	81.17	80.56	80.92
Athlete 14	69.54	71.14	72.34	70.16	72.01	71.11
Athlete 15	76.01	77.42	77.41	78.32	78.22	77.61
Athlete 16	68.31	70.11	70.05	70.17	69.02	70.06
Athlete 17	58.53	60.04	60.56	59.69	59.72	60.69
Athlete 18	81.64	82.61	83.30	83.66	82.87	82.60
Athlete 19	55.03	56.70	56.35	56.90	56.48	56.37
Athlete 20	65.03	66.33	66.44	67.29	67.28	66.82

$$OS = 1 + (1.01 \times TS) + \mathcal{N}(0, 0.5) \quad (10.3)$$

Table 10.1 contains the simulated sample for N=20 athletes and 5 trials.

The objective of the analysis is to estimate DGP parameters of the measurement error (the proportional bias, fixed bias, and the SD of the random error). Unfortunately, since TS is unknown, we are unable to estimate proportional bias and fixed bias. To overcome this problem, we usually compare OS to some *gold standard* (or *criterion*) measure which can serve as proxy to TS. These issues are covered in much more detail in the second part of this book in the [Validity and Reliability](#) chapter.

What is left to be estimated is the SD of the random error, which is often referred to as *typical error* (TE) of the test, or *standard error of the measurement* (SEM)². TE is estimated using individual SD of the OS in the five trials (Table 10.2).

The **mean** of athletes' typical errors (SD in Table 10.2) is equal to 0.45kg, which is quite close to DGP random error parameter of 0.5kg. The reason for the

²Both SEM and TE are used in the sports science literature and research. I personally prefer the simple *random error*, followed by TE. SEM has the same meaning as the *standard error of the mean* explained in the [Frequentist perspective](#) chapter and might create confusion.

Table 10.2: Individual mean and SD from five trials

Athlete	Mean	SD
Athlete 01	79.59	0.45
Athlete 02	77.60	0.57
Athlete 03	78.46	0.16
Athlete 04	56.33	0.21
Athlete 05	85.63	0.40
Athlete 06	62.91	0.78
Athlete 07	70.34	0.37
Athlete 08	62.48	0.42
Athlete 09	82.38	0.63
Athlete 10	93.54	0.28
Athlete 11	81.73	0.39
Athlete 12	68.62	0.75
Athlete 13	80.82	0.24
Athlete 14	71.35	0.86
Athlete 15	77.80	0.44
Athlete 16	69.88	0.49
Athlete 17	60.14	0.47
Athlete 18	83.01	0.46
Athlete 19	56.56	0.24
Athlete 20	66.83	0.45

Table 10.3: **Estimating Typical Error using SD of the difference scores**

Athlete	OS 1 (kg)	OS 2 (kg)	Difference OS 2-1 (kg)
Athlete 01	79.47	79.37	-0.11
Athlete 02	77.83	77.71	-0.11
Athlete 03	78.65	78.30	-0.35
Athlete 04	56.58	56.36	-0.23
Athlete 05	85.53	85.44	-0.09
Athlete 06	61.84	63.05	1.22
Athlete 07	70.26	69.82	-0.44
Athlete 08	62.03	62.91	0.88
Athlete 09	81.34	83.04	1.70
Athlete 10	94.02	93.47	-0.55
Athlete 11	81.89	82.13	0.24
Athlete 12	69.54	68.32	-1.22
Athlete 13	80.66	80.81	0.15
Athlete 14	71.14	72.34	1.21
Athlete 15	77.42	77.41	-0.01
Athlete 16	70.11	70.05	-0.06
Athlete 17	60.04	60.56	0.52
Athlete 18	82.61	83.30	0.70
Athlete 19	56.70	56.35	-0.35
Athlete 20	66.33	66.44	0.11

difference between estimated and true value of the random error SD is due to the *sampling error*, which is a topic covered in the [Statistical inference](#) section of this book.

Unfortunately, this method of estimating TE is not always practically feasible. TE is usually estimated with two trials (OS1 and OS2; see Table 10.3), by using SD of the difference scores (SD_{diff}) across athletes (Hopkins 2000; Swinton et al. 2018).

If we calculate SD of the difference scores from the Table 10.3, we get 0.7kg. However, this is not quite right, since we know that the true SD of the random error is 0.5kg. This happens because random error is affecting both Trial 1 (OS1) and Trial 2 (OS2), and is *propagated* to the difference between the two (Figure measurement-error-propagation). This is exactly the same concept as described in the [Example of randomized control trial](#). The benefit of using squared errors and assuming Gaussian error distribution, as alluded multiple times in this book, is that this propagation can be mathematically and neatly expressed and estimated.

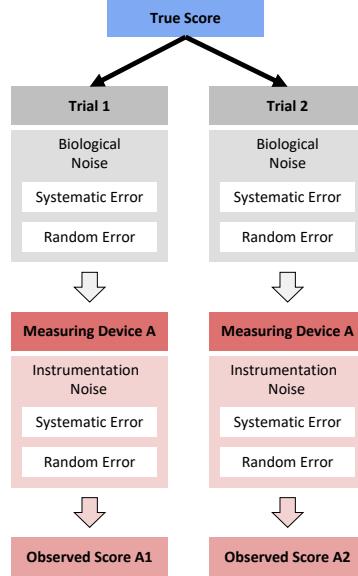


Figure 10.2: Propagation of the random component of measurement error to two trials

Error propagation, assuming normal distribution, is equal to Equation (10.4):

$$\begin{aligned}
 SD_{difference}^2 &= TE_{trial 1}^2 + TE_{trial 2}^2 \\
 TE_{test} &= TE_{trial 1} = TE_{trial 2} \\
 SD_{difference}^2 &= TE_{test}^2 + TE_{test}^2 \\
 SD_{difference}^2 &= 2 \times TE_{test}^2 \\
 SD_{difference} &= \sqrt{2 \times TE_{test}^2} \\
 SD_{difference} &= \sqrt{2} \times TE_{test} \\
 TE_{test} &= \frac{SD_{difference}}{\sqrt{2}}
 \end{aligned} \tag{10.4}$$

According to the Equation (10.4), the SD of the difference score needs to be divided by $\sqrt{2}$ to estimate TE of the measurement. The assumption is that TE is equal in both trials (and for all athletes), which is defined in the second line in the Equation (10.4). Estimated TE is now equal to 0.49kg, which is much closer to known SD of the random error of 0.5kg.

10.1 Estimating TE using *ordinary least products* regression

The method of estimating TE using SD of the difference score can be termed *method of the differences* and it is quite commonly used in [Validity and Reliability](#) analyses, particularly when using Bland-Altman plots. Another method involves the use of the linear regression, or the special kind of linear regression called *ordinary least products* (OLP) (Ludbrook 2010, 2012, 1997, 2002; Mullineaux, Barnes, and Batterham 1999). The *ordinary least squares* (OLS; the one we have used thorough this book) regression find the line that minimizes squared residuals between y_i and \hat{y}_i , while OLP minimizes the product of the residuals using both x and y : $(x_i - \hat{x}_i) \times (y_i - \hat{y}_i)$. The benefit of using OLP over OLS is that estimated model is the same regardless of which variable is considered outcome or predictor. This is not the case with OLS, as demonstrated in the [Describing relationship between two variables](#) section. For this reason, OLP is a preferred choice when performing Reliability analyses, since neither variable is considered outcome.

Figure 10.3 demonstrate OLP regression between Trial 1 (OS1) and Trial 2 (OS2). Estimated residual standard error ([RSE](#)) is equal to 0.72cm. To estimate TE, this [RSE](#) also needs to be divided by $\sqrt{2}$, which results in 0.51cm. This estimate of TE is very close to TE estimated by the method of differences.

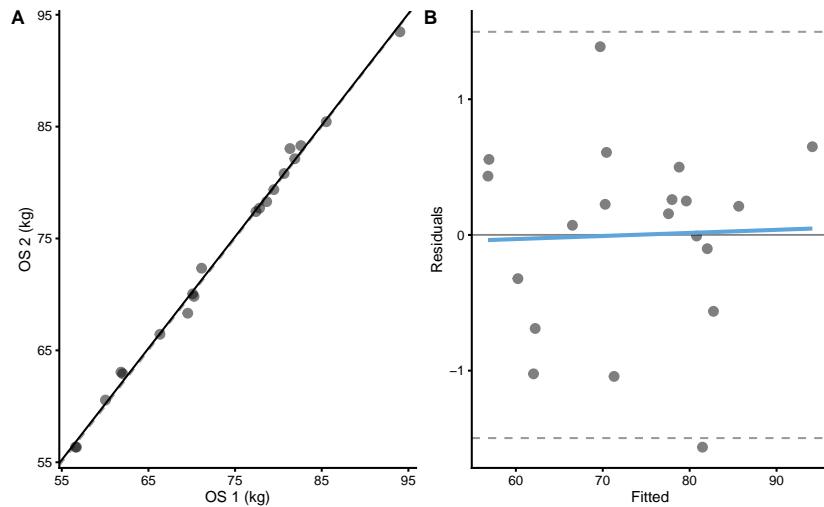


Figure 10.3: Ordinary least squares regression (OLP) between Trial 1 (OS1) and Trial 2 (OS2). **A.** Scatter-plot between OS1 and OS2. Dashed line represent identity line, and black line represent estimated OLP regression. **B.** Residuals plot. Dashed lines represent upper and lower *levels of agreement* using RSE and 95% confidence level (or in other words, 95% of the residuals distribution will be within these two dashed lines). Blue line represent additional linear regression model (using OLS) between fitted and residual, used to indicate issue with the model.

If we consider $\pm 1\text{kg}$ to be SESOI in the observed score, we can estimate practical reliability of this scale. Magnitude-based estimators, such as PPER or SESOI to RSE can be used to quantify scale reliability from a practical significance perspective. This can be represented with the SESOI band as done in the Figure 10.4.

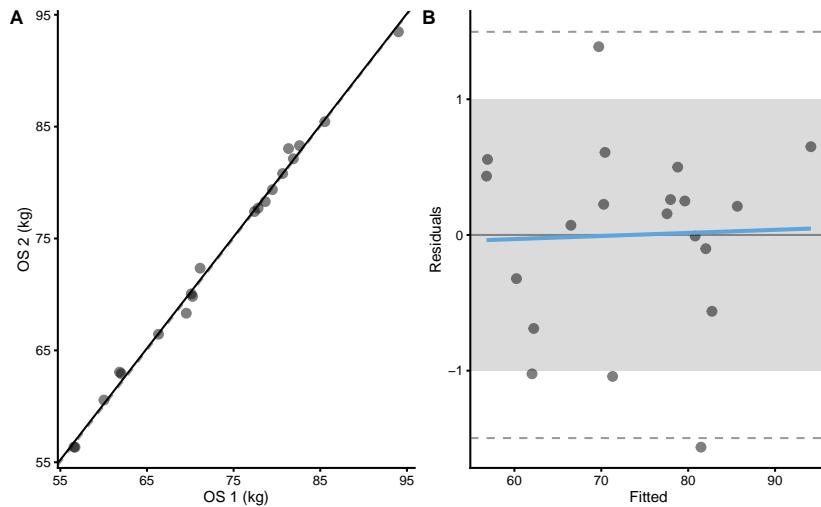


Figure 10.4: Ordinary least squares regression (OLP) between Trial 1 (OS1) and Trial 2 (OS2). **A.** Scatter-plot between OS1 and OS2. Dashed line represent identity line, and black line represent estimated OLP regression. **B.** Residuals plot. Dashed lines represent upper and lower *levels of agreement* using RSE and 95% confidence level (or in other words, 95% of the residuals distribution will be within these two dashed lines). Blue line represent additional linear regression model (using OLS) between fitted and residual, used to indicate issue with the model. SESOI is represented with the grey band. Residuals within SESOI band are of no practical difference. Proportion of residuals within SESOI band represent PPER estimator.

As can be seen from the Figure 10.4, this scale have less than perfect reliability to detect practically important changes in weight (given *a priori* defined SESOI of $\pm 1\text{kg}$). Estimated PPER, using RSE, is equal to 0.82.

10.2 Smallest Detectable Change

In the case of reliability and validity analysis, we are mostly interested in the ability of a measure to detect signal from a noise. In this case, the *signal* is the true or real change and *noise* is the random error of the measure. Thus, we are interested in estimating what is the smallest detectable signal the measure can detect with certain level of confidence. This is called *smallest detectable change* (SDC) or *minimal detectable change* (MDC).

If you check the Panel B in the Figure 10.4, SDC represents the spread of the residuals, visualized with the two horizontal dashed lines. If we assume that the residuals are normally distributed, we can estimate lower and upper thresholds

that capture, for example 95% of the residuals distribution. This is done by multiplying SD_{diff} or RSE with appropriate critical value from a Student's t-distribution (i.e. or simply with ± 1.96). We thus get the Equation (10.5) that we can use to estimate SDC with 95% level of confidence³.

$$\begin{aligned} SDC &= SD_{diff} \times \pm 1.96 \\ SDC &= RSE \times \pm 1.96 \\ SDC &= TE \times \sqrt{2} \times \pm 1.96 \end{aligned} \quad (10.5)$$

Using OLP regression estimate RSE (equal to 0.72), and critical value to cover 95% of the Student's t-distribution ($DF=19$) equal to ± 2.09 , SDC for our scale is equal to $\pm 1.5\text{kg}$. This implies that, with the 95% confidence, we are able to detect true signal (i.e. change in weight) as low as $\pm 1.5\text{kg}$. If SDC is lower than SESOI, reliability of the measure is (practically) perfect. This is not the case for our scale. We cannot use it to detect changes of $\pm 1\text{kg}$ with satisfying level of confidence.

SDC can also be used as SESOI in some other analysis utilizing this scale. For example, if we use nutrition intervention RCT utilizing this scale as a measure, we can use $\pm 1.5\text{kg}$ as SESOI (in the absence of better defined SESOI) since that is the the minimum detectable effect size.

10.3 Interpreting individual changes using SESOI and SDC

In order to showcase the interpretation of the individual changes by using SESOI and SDC (named *observed outcome approach* in [Analysis of the individual residuals: responders vs non-responders](#) section), let's consider bench press example from the [Comparing dependent groups](#) (repeated in the Table 10.4).

Before commencing this simple intervention, measurement error of the bench press 1RM test is estimated (using TE estimator), and is equal to 2.5kg ($N=19$). Practically, this means that due to the biological and instrumentation error, 1RM in the bench press would tend to vary normally distributed with TE equal to 2.5kg , given, of course, no *real* change in the strength (i.e. no change in TS). Expected SDC (with 95% confidence level) is thus equal to $\sqrt{2} \times TE \times \pm 2.09$ ($\pm 7.39\text{kg}$). Please note that TE of the *change score* is equal to $\sqrt{2} \times TE$, or (3.54kg).

For the sake of example, $\pm 5\text{kg}$ can be considered minimal important change, which will be used as SESOI. Since both TE and SESOI are known, the objective

³This should also be done with TE if we want to quantify uncertainty around single observation with a particular level of confidence (e.g. multiply TE with ± 1.96 to get 95% confidence), assuming normally distributed random error that is additive.

10.3. INTERPRETING INDIVIDUAL CHANGES USING SESOI AND SDC181

Table 10.4: Individual Pre and Post scores, as well as Change in the bench press 1RM

Athlete	Pre-test (kg)	Post-test (kg)	Change (kg)
Athlete 01	111.80	121.42	9.62
Athlete 02	95.95	102.13	6.18
Athlete 03	105.87	125.56	19.69
Athlete 04	98.79	109.67	10.87
Athlete 05	95.81	108.11	12.30
Athlete 06	95.27	92.67	-2.60
Athlete 07	97.75	106.03	8.28
Athlete 08	106.50	109.51	3.01
Athlete 09	80.62	95.96	15.34
Athlete 10	100.40	94.30	-6.11
Athlete 11	82.71	78.91	-3.80
Athlete 12	102.89	93.98	-8.91
Athlete 13	91.34	105.21	13.87
Athlete 14	111.14	108.07	-3.07
Athlete 15	95.13	96.01	0.88
Athlete 16	109.12	112.12	3.00
Athlete 17	91.87	103.41	11.54
Athlete 18	92.16	103.93	11.77
Athlete 19	108.88	119.72	10.84
Athlete 20	97.94	95.91	-2.03

of the analysis is to estimate probability that the observed individual change score is practically significant (i.e. lower, equivalent, or higher compared to SESOI). This is because individual *true changes* are not known, but only *observed changes*. Change TE tells how much of observed change randomly varies around the true change score. The question trying to be answered is: "how likely individual's *true change* is within lower, equivalent, or higher SESOI range, given the known change TE?"

Panel A in the Figure 10.5 depicts individual Change scores *probabilistically* using the known change TE (3.54kg). Using the SESOI as equivalent change, we can estimate individual probability of lower, equivalent, and higher change. Panel B in the Figure 10.5 depicts individual change scores with error bars representing SDC. The numbers in brackets on Panel B in the Figure 10.5 represent estimated probabilities of the true change score being lower, equivalent, and higher compared to SESOI. To be more certain of individual changes, SDC needs to be smaller compared to SESOI. Ratio between SESOI and change TE can thus represent an estimate of the *test sensitivity* to detect practically meaningful changes (i.e. SESOI to RSE estimator). The smallest change that has at least 95% chance of being higher or lower than SESOI is equal to $SESOI \pm SDC$, or $\pm 12.39\text{kg}$. Graphically, bench press 1RM change of $\pm 12.39\text{kg}$ is the smallest change, where 95% confidence intervals do not touch the SESOI band. Please note that inference from MET is slightly different, since METs use single-tailed tests, thus the critical value will be smaller than 2.09 (it will be 1.73 for single-tailed 95% confidence). This implies that 95% confidence intervals (i.e. SDC) can slightly cross SESOI threshold and still be considered "Higher" or "Lower."

10.3. INTERPRETING INDIVIDUAL CHANGES USING SESOI AND SDC183

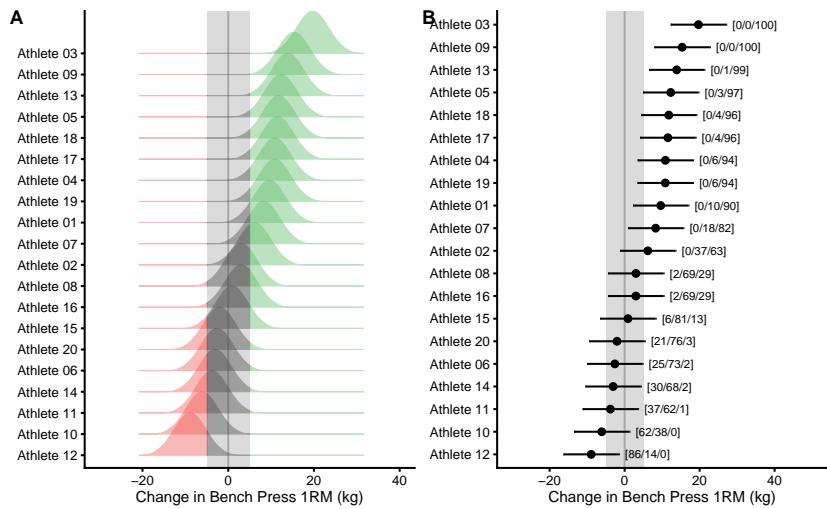


Figure 10.5: **Analysis of individual change scores using SESOI and SDC.** **A.** Uncertainty around true change score can be depicted by using normal distribution whose SD is equal to change TE. Probability that the observed change is lower, equivalent or higher can be estimated using surface within lower, equivalent, and higher magnitude band. **B.** 95% Confidence intervals around change scores represent SDC and are calculated using $\pm 2.09 \times \sqrt{2} \times TE$. Numbers in brackets represent proportion of the surface area in the lower, equivalent and higher magnitude band. These are interpreted as probabilities of true score being in lower, equivalent and higher magnitude band. See text for discussion why such interpretation is not statistically valid

As explained in the [Statistical inference](#) section of this book, this method of individual analysis interprets the change TE and associated confidence intervals from the Bayesian perspective. This is not the correct interpretation, since we do not know individual's true change scores, only the observed change scores. Change TE gives us the variance of the observed change scores around the true change score, not *vice versa* (i.e. Bayesian *inverse probability*). Thus, visual representation from the Figure 10.5 is not statistically valid.

Since we do not know the true change scores, we are interested in probabilities of seeing the observed change score given the assumption of where we think the true change score is (i.e. null hypothesis). For this reason the question to be asked is “assuming individual’s true change scores are at \pm SESOI, how likely are we to see the observed change score, given the known change TE?”. This question demands answer and interpretation of change TE from the frequentist perspective. Thus the correct interpretation of the individual changes involves the use of minimum effect tests (METs) discussed in the [Statistical Inference](#) section. METs approach to interpreting individual changes is depicted in the Figure 10.6.

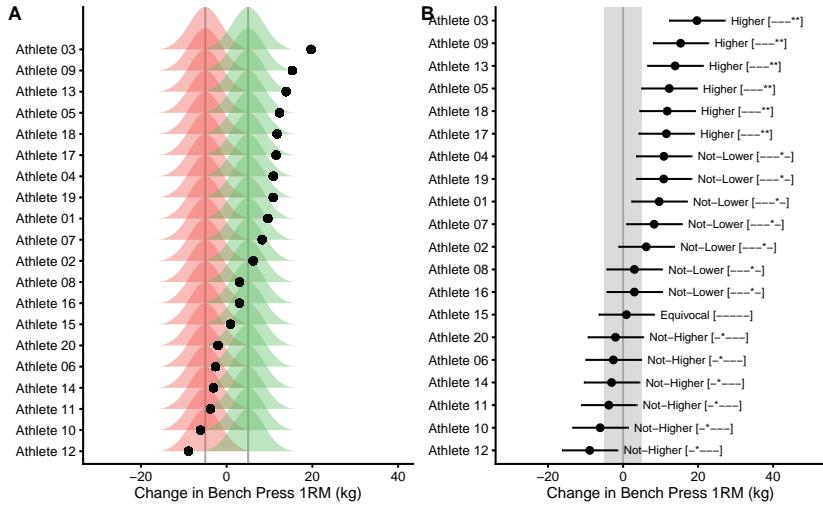


Figure 10.6: METs approach to interpreting individual change scores.

A. Since true change scores are unknown, we can only test probability of seeing observed score or higher, given the known TE and assumed true score. In order to do this, minimal effect tests are performed, assuming two true score null-hypotheses: one at lower SESOI threshold (red color) and one at upper SESOI threshold (green color). Typical error can be interpreted as SD of the error distribution. Black dots indicate observed individual change. We are interested in estimating probability of observing this change, given two hypotheses. Five tests are performed as described in Minimum Effect Tests section: inferiority, non-superiority, equivalence, non-inferiority and superiority. **B.** 95% Confidence intervals around change scores represent SDC and are calculated using $\pm 2.09 \times \sqrt{2} \times TE$ and depicted using error-bars. Final inference using five METs is reported. METs significance (assuming alpha=0.05), indicated by '*', are reported in the brackets, for each of the five tests performed

10.4 What to do when we know the error?

Statistical analysis covered in this book treats observed scores as true scores without measurement error. But measurement error is always involved and can introduce bias in the conclusion. How certain estimators and analyses are sensitive to measurement error can be estimated via simulations. But what can we do when we do know that there is measurement error involved in our observations and we actually have an estimate about its size (i.e. from validity or reliability study)?

There are few *adjustment* techniques that can be implemented (Keogh et al. 2020; Lederer and Küchenhoff 2006; Shang 2012; Shaw et al. 2020; Wallace 2020), but here I will introduce *Simulation extrapolation* (SIMEX) approach

since it is very intuitive. Let's take the bench press example we used above: we do know that measurement error of 2.5kg is involved in the observed Pre- and Post-tests. How can we *correct* or *adjust* our estimators using that knowledge?

Since we know that error is already inside our observations, we can call that error factor or error multiplier of 1. Then we add additional error to our observations and repeat the analysis. This is done for error multipliers from 1 to usually 3 (i.e. extra $2 \times$ measurement error). Let's do that using bench press data and calculate mean and SD of the Pre-test, Post-test, and Change. This single simulation is in Figure 10.7.

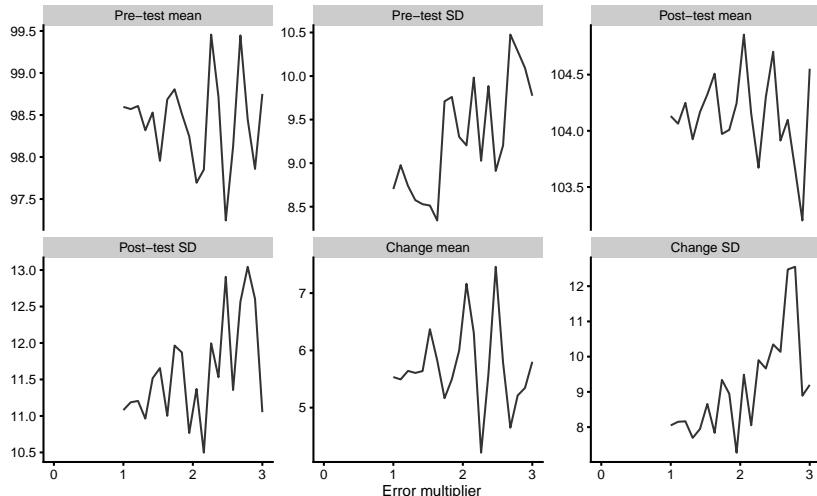


Figure 10.7: Adding noise (measurement error) to observed Pre-test and Post-test scores and re-calculating estimators. Error multiplier equal to 1 is *naive* analysis where one magnitude of measurement error (i.e. 2.5kg) is already involved in observed data. Additional error is added using error multiplier (i.e. error multiplier 2 involves additional noise of 2.5 kg magnitude, thus 2 error magnitudes are involved in the data) from 1 to 3, using total of 20 steps

We can't conclude much since adding error multiplier once is *stochastic*. We need to repeat this numerous times, say 100 times. This is depicted in Figure 10.8.

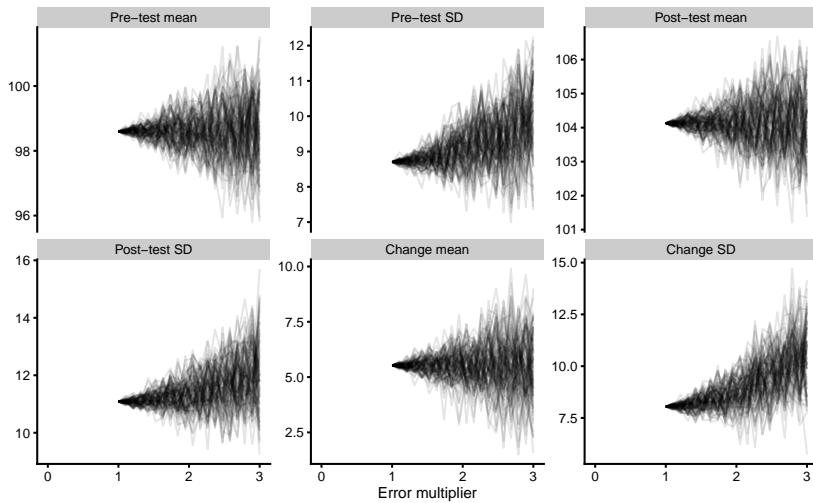


Figure 10.8: Result of SIMEX using 100 simulations

What we are interested in, is calculating the average or expected estimator value for each error multiplier. This is depicted in Figure 10.9

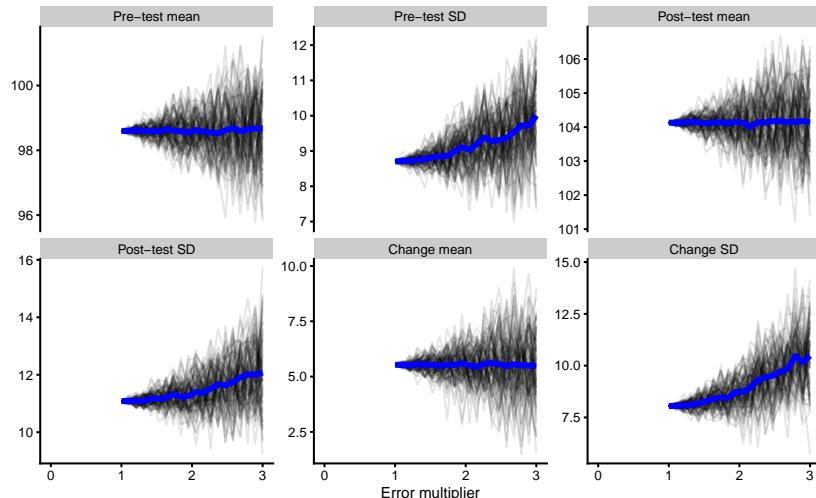


Figure 10.9: Result of SIMEX using 100 simulations and addition simulation average. Blue line represents simulations average for a particular error multiplier

Rather than using average across simulations, we can fit a particular model and then *extrapolate* to error multiplier of 0. This way we can get estimated

estimator value when there is no measurement error involved in Pre-test and Post-test variables. Usually this is done using second order polynomial (i.e. $\hat{y}_i = \beta_0 + \beta_1 x_i + \beta_2 x_i^2$), or quadratic equation (i.e. $\hat{y}_i = \beta_0 + \beta_1 x_i^2$). Extrapolating using quadratic equation is depicted in the Figure 10.10.

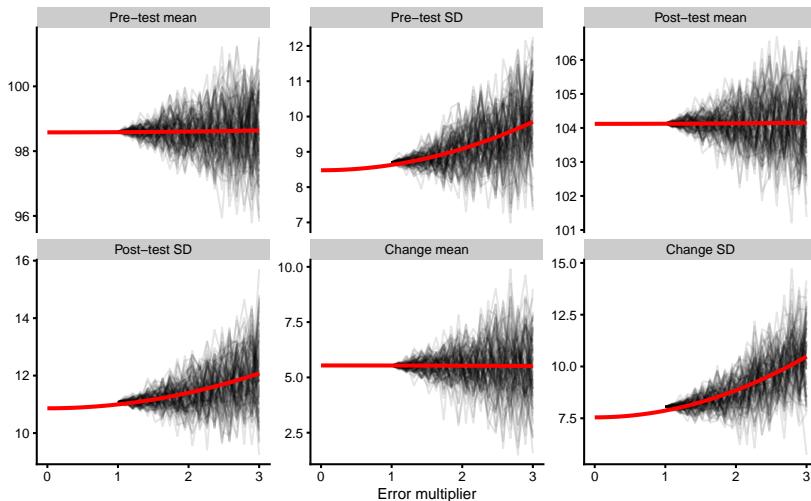


Figure 10.10: Result of SIMEX using 100 simulations and addition quadratic extrapolation. Red line represents quadratic fit, extrapolated to error multiplier of 0 to estimate estimator value when there is no measurement error involved in the Pre-test and Post-test variables

From 10.10 it is interesting to notice that **mean** is *robust* to measurement error, while **SD** is not, which is expected since normal error increase *variance* of the data. Using SIMEX for **Change SD**, we can estimate *true* Change SD, or in other words, when there is no measurement error. This can also be done using math rather than simulation and extrapolation (i.e. SIMEX) by using the same error propagation reasoning explained in the Example of randomized control trial section (Equation (10.6)).

$$\begin{aligned}\sigma_{OS}^2 &= \sigma_{TS}^2 + \epsilon^2 \\ SD_{observed\ diff}^2 &= SD_{true\ diff}^2 + (\sqrt{2}TE)^2 \\ SD_{true\ diff} &= \sqrt{SD_{observed\ diff}^2 - 2TE^2}\end{aligned}\tag{10.6}$$

Observed Change SD is equal to 8.05kg, while the change TE is equal to $\sqrt{2} \times 2.5\text{kg}$ or 3.53kg. True Change SD is thus equal to $\sqrt{8.05^2 - 3.53^2}$, or 7.23kg. This is also done when estimating stochastic treatment effect in the RCT, if we assume that the Change SD in the Control group is mainly due to measurement error.

10.5 Extending the Classical Test Theory

The theory behind true, observed scores, and measurement error introduced in this chapter is called Classical Test Theory (Borsboom 2009). Although it sounds simple, there are numerous assumptions and issues with it (Borsboom 2009, 44–45):

"Classical test theory was either one of the best ideas in twentieth-century psychology, or one of the worst mistakes. The theory is mathematically elegant and conceptually simple, and in terms of its acceptance by psychologists, it is a psychometric success story. However, as is typical of popular statistical procedures, classical test theory is prone to misinterpretation. One reason for this is the terminology used: if a competition for the misnomer of the century existed, the term 'true score' would be a serious contestant. The infelicitous use of the adjective 'true' invites the mistaken idea that the true score on a test must somehow be identical to the 'real', 'valid', or 'construct' score. This chapter has hopefully proved the inadequacy of this view beyond reasonable doubt.

The problems with the platonic true score interpretation were, however, seen to run deeper than a confounding of validity and unreliability. Classical test theory is, ontologically speaking, ambiguous. In principle, it allows for both realist and constructivist interpretations, but sits well with neither. The constructivist interpretation of classical test theory is vacuous: although it is possible to specify how a true score could be constructed on the basis of observations (namely by averaging), the observations that are necessary for doing this are exactly the repeated measurements with intermediate brainwashing. The constructivist account of true scores can only be interpreted metaphorically, and it is not at all clear what such a metaphorical interpretation adds to the theory. On the other hand, a realist interpretation of true scores leads to a metaphysical explosion of reality: a new true score has to be invoked for every thinkable testing procedure."

Let's take the bench press example yet again. What is individual's *true score*? Something that one can manifest? What if I have bench pressed 150kg, but got a flu and was off for 3 weeks. When I came back, I was able to bench press 130kg. Is my *true* bench press still 150kg, but *masked* due to systematic effect of the illness? What if this happens in a reliability study, and few individuals demonstrate true systematic effects of fatigue, while some demonstrate biological variability from day to day? We do assume *Ergodicity* in this example.

Figure 10.11 depicts my extension of the Classical Test Theory with performance specialist or sports scientist in mind (Jovanović 2020). As alluded multiple

time thorough this book, all statistical models represent “Small Worlds”, or simplifications of the complex “Large World” we ought to understand and interact with.

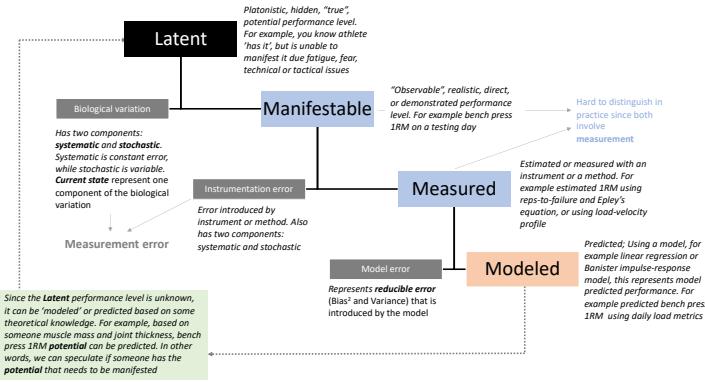


Figure 10.11: **Circular Performance Model.** Extended Classical Test Theory to include *phenomenology* of working with athletes

Circular Performance Model depicted in the Figure 10.11 (Jovanović 2020) can be used to model and understand *phenomena* that sport practitioners and sports scientists wrestle with daily.

Chapter 11

Conclusion

Statistical analysis should start with questions that we are trying to answer using the data. These questions of interest should not only guide statistical analysis, but also guide data collection and finally interpretations of the analysis and modelling results. In order to answer these questions with data, we are always representing the Large World with the Small World models. There is no entirely objective approach to do it, rather pluralism of approaches (Mitchell 2012, 2002) should be applied. The value of these models and Small World representations should be judged by qualities suggested by Gelman and Hennig (Gelman and Hennig 2017): *transparency, consensus, impartiality, correspondence to observable reality, awareness of multiple perspectives, awareness of context-dependence, and investigation of stability*. Finally, we need to accept that we must act based on *cumulative knowledge* rather than solely rely on single studies or even single lines of research (Amrhein, Trafimow, and Greenland 2019).

For example, let's take a question that a sport practitioner might ask: "From how many athletes I can expect to see positive improvements after this intervention? Will Johnny improve?" This question is predictive question, which is common in practice. Assume that I provide this coach with the estimate of the average causal effect and accompanying magnitude-based inference (MBI), using SESOI he provided (i.e. 20% harmful, 30% equivalent, and 50% beneficial) or frequentist p-value of $p < 0.05$. Will that answer the practitioner's question?

The accompanying MBIs might even confuse him and appear to answer the "proportion" question he asked. "So, 50% of athletes will show beneficial response to treatment?". Unfortunately no - MBIs (or METs) answer different question about estimator (be it `mean` or Cohen's `d` or some other), not about individual response proportions.

Second part of his question also demands predictive modeling, that calls for taking into account Johnny's known data and getting the best estimate for his response. If there is some historical data about Johnny, we might get better

predictions (i.e. either through individual modeling or *hierarchical model*), but if not, then the average-based estimators might be our best guess of the most likely response that Johnny might manifest. Reporting proportions of responses on top of the average effect estimate might help answering questions about uncertainty regarding individual responses.

I am not saying that these are not important. I am only saying that they should not be automatically selected as an answer to any question. CIs can provide us with the uncertainty interval around proportions (i.e. “Model gives us 90% confidence that the proportion of the beneficial responses will be 40-60%”), or even around predictive performance metrics. However, we need to make sure to start with the question asked, as well as to suit our analysis and conclusions so that practitioners can understand it, and finally act based on it.

In the following part of this book, I will provide solution to the most common sport science problems and question using the material covered in this part, as well as **bmbstats** package written by the author.

Part II

Part Two

Chapter 12

bmbstats: Bootstrap Magnitude-based Statistics package

In the first part of this book we have covered descriptive, predictive, and causal inference tasks, followed by the basics of statistical inference using frequentist, Bayesian, and bootstrap methods and concluded with the measurement error explanation. In this part of the book, we will turn to **bmbstats**, which is short of *Bootstrap Magnitude-based Statistics*, package to perform analysis of the most common sports science problems and utilize bootstrap as the inference method of choice (Jovanović 2020a).

Since I strongly believe that the best way to understand statistical analysis is through simulations and smart visualizations, in this part of the book I will show the R code that you can use to reproduce the analysis, but more importantly, to understand the underlying DGP we are trying to explain (or estimate) with the analysis.

12.1 bmbstats Installation

You can install the development version from GitHub¹ with:

```
# install.packages("devtools")
devtools::install_github("mladenjovanovic/bmbstats")

require(bmbstats)
```

¹<https://github.com/mladenjovanovic/bmbstats>

Chapter 13

Descriptive tasks using **bmbstats**

In this chapter I will demonstrate the analysis of the basic descriptive tasks using **bmbstats** package. Each data set that we will analyze will be generated with the R code, so the underlying DGP will be transparent and thus very useful for understanding what we are trying to do with the analysis.

13.1 Generating *height data*

In [Comparing two independent groups](#) chapter we have used height data for 50 males and 50 females. This is how that data is generated:

```
require(bmbstats)
require(tidyverse)

set.seed(1667)

n_subjects <- 50

# Generate height data
height_data <- data.frame(Male = rnorm(n = n_subjects, mean = 177.8,
                                         sd = 10.16), Female = rnorm(n = n_subjects, mean = 165.1,
                                         sd = 8.89))

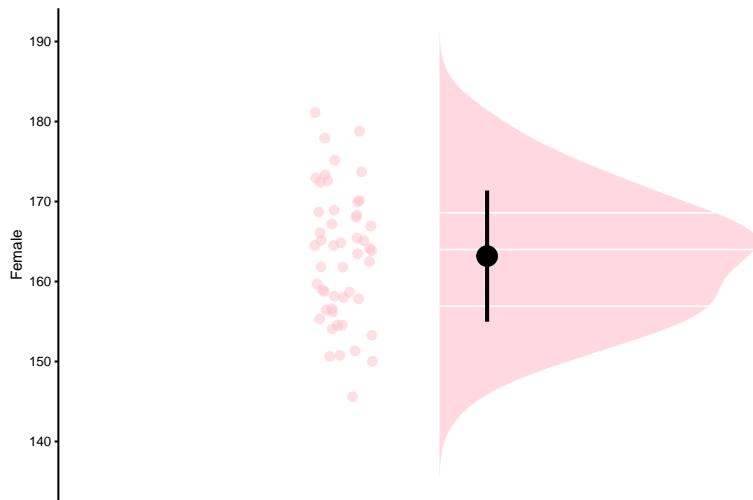
head(height_data)
#>      Male   Female
#> 1 193.9007 150.7703
```

```
#> 2 172.4291 150.0221
#> 3 186.3210 170.1512
#> 4 177.4417 156.4366
#> 5 167.5636 156.1961
#> 6 172.9078 158.9467
```

13.2 Visualization and analysis of a single group/variable

The simplest descriptive task is the description of a single group. Let's use height of the females as an example. Function `bmbstats::plot_raincloud`¹ can be used to plot the distribution and summary statistics (mean and SD as error bar):

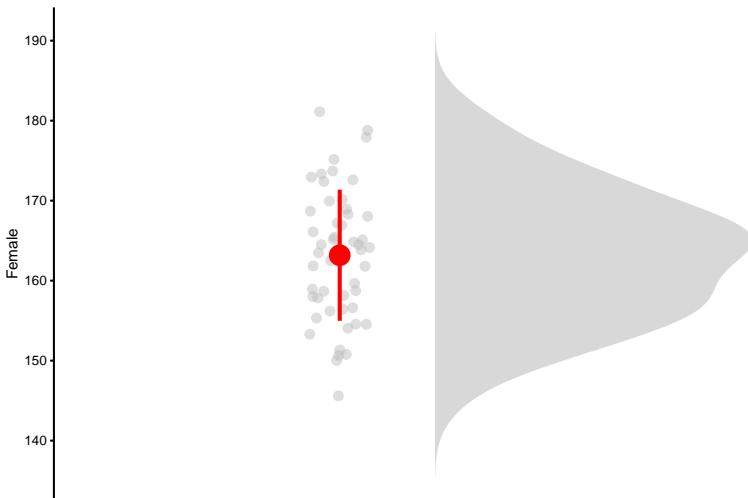
```
bmbstats::plot_raincloud(data = height_data, value = "Female",
control = plot_control(group_colors = "pink"))
```



Functions in `bmbstats` package use `control` parameter to setup graphing or modeling parameters. For example, we can remove the quantile lines, resize points, change color by using `bmbstats::plot_control` function in the `control` parameter:

¹I will refer to `bmbstats` functions using prefix `bmbstats::` although you can use those functions without it

```
bmbstats::plot_raincloud(data = height_data, value = "Female",
  control = plot_control(group_colors = "grey", cloud_quantile_lines = FALSE,
    summary_bar_color = "red", summary_bar_nudge = -0.15,
    points_jitter_width = 0.1, points_size = 2))
```



One of the core functions in `bmbstats` package is `bmbstats::bmbstats`, around which multiple *wrapper* functions are built, such as `bmbstats::describe_data`. `bmbstats::describe_data` performs bootstrap using the estimators provided in the `estimator_function` parameter. To modify bootstrap parameters, use `control` parameter and `bmbstats::model_control` function:

```
female_analysis <- bmbstats::describe_data(x = height_data$Female,
  estimator_function = bmbstats::data_estimators_simple,
  control = model_control(seed = 1667, boot_type = "perc",
    boot_samples = 1000, confidence = 0.9))

female_analysis
#> Bootstrap with 1000 resamples and 90% perc confidence intervals.
#>
#>   estimator      value      lower      upper
#>     mean 163.177916 161.200476 165.143065
#>     SD   8.199373  6.914061  9.136136
```

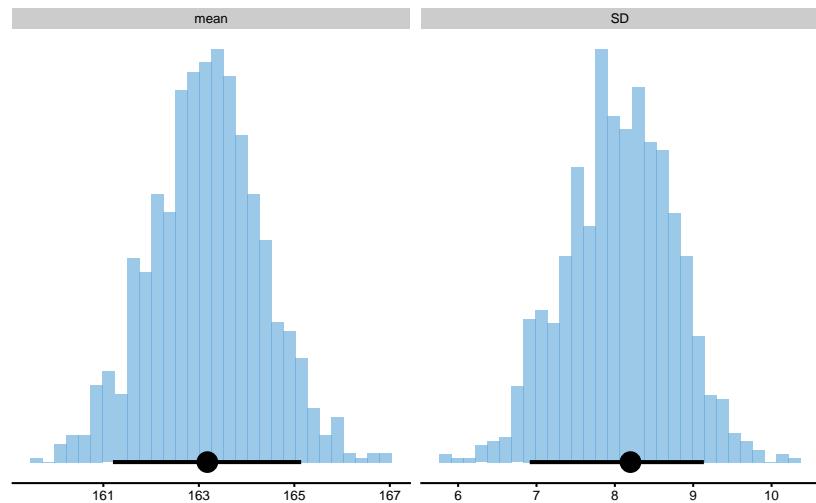
The above code uses `bmbstats::data_estimators_simple` function that returns `mean` and `SD` estimators. If you want to access the *data frame* containing estimators values and upper and lower confidence thresholds use the following code:

```
female_analysis$estimators
```

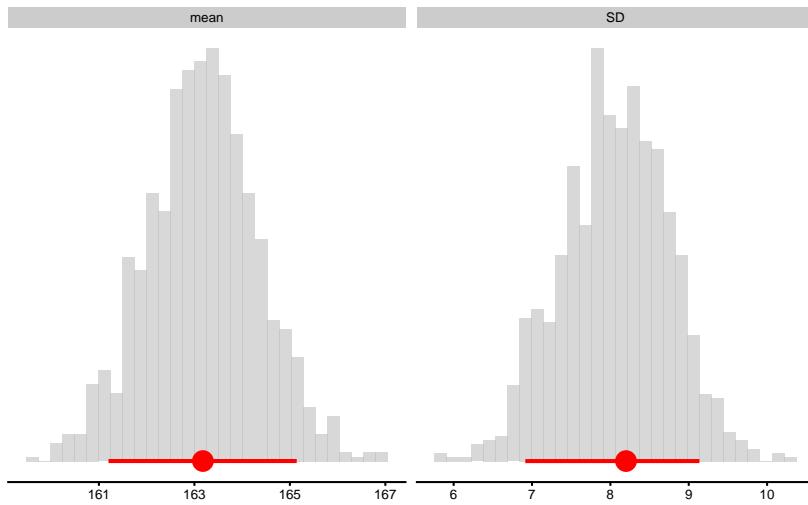
Since we have generated the data, we know the *true* population parameters for the **mean** (165.1cm) and **SD** (8.89cm), and we are hoping that our 90% confidence intervals capture those values in 90% of the cases in the long run (i.e. have 10% Type I error).

To plot bootstrap distributions, use simple `plot` function:

```
plot(female_analysis)
```



The figure above depicts distribution of the bootstrap resamples with the error bar representing estimator value and upper and lower confidence thresholds (in this case 90% estimated using *percentile* method). To change the colors, use `control` and `plot_control`:



13.2.1 Using your own estimators

`bmbstats` functions are *modular*, implying that you can use different modules that you write yourself. This includes `estimators` function, but also performance functions (covered later when discussing prediction using `bmbstats`). Let's say we are interested in the calculating `mean`, `SD`, and `CV%` (coefficient of variation):

```
my_estimators <- function(x, na.rm = FALSE) {
  x_mean <- mean(x, na.rm = na.rm)
  x_sd <- sd(x, na.rm = na.rm)

  x_cv <- (x_sd/x_mean) * 100

  return(c(mean = x_mean, SD = x_sd, CV = x_cv))
}
```

If we apply this function to female heights, we get the following estimates:

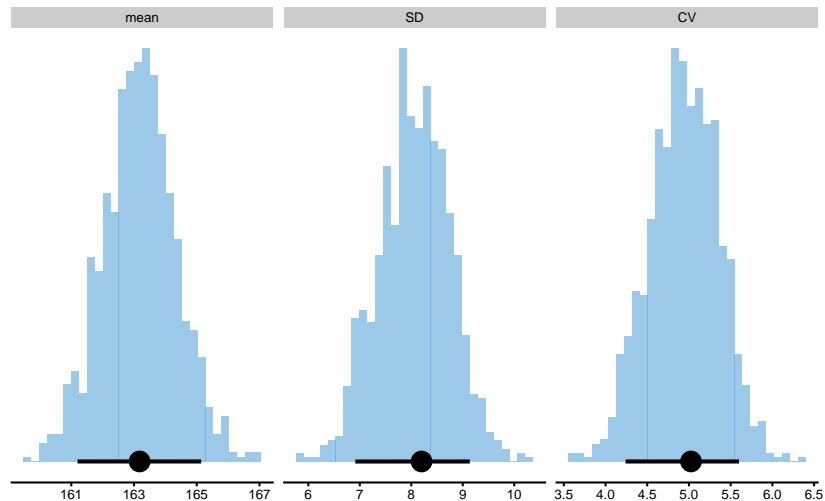
```
my_estimators(height_data$Female)
#>      mean        SD        CV
#> 163.177916   8.199373   5.024806
```

Since we are interested in making statistical inference (by utilizing bootstrap method), we can simple replace `bmbstats::data_estimators_simple` with `my_estimators`:

```
female_analysis_my_est <- bmbstats::describe_data(x = height_data$Female,
  estimator_function = my_estimators, control = model_control(seed = 1667,
  boot_type = "perc", boot_samples = 1000, confidence = 0.9))
```

```
female_analysis_my_est
#> Bootstrap with 1000 resamples and 90% perc confidence intervals.
#>
#>   estimator      value      lower      upper
#>   mean  163.177916 161.200476 165.143065
#>   SD    8.199373  6.914061  9.136136
#>   CV    5.024806  4.240318  5.600490
```

```
plot(female_analysis_my_est)
```



`bmbstats::describe_data` comes with three estimator functions: `bmbstats::data_estimators`, `bmbstats::data_estimators_simple`, and `bmbstats::data_estimators_robust`.

Let's run the `bmbstats::data_estimators` and `bmbstats::data_estimators_robust`, but this time using `bca` method of estimating 95% bootstrap confidence intervals (CIs):

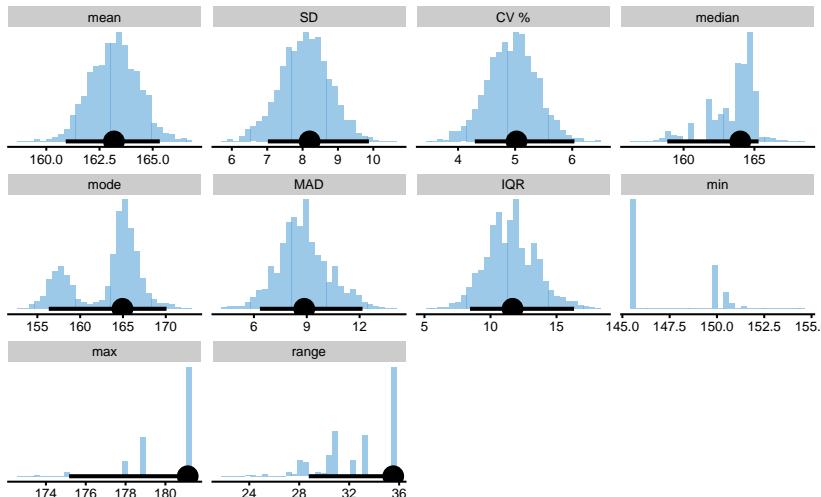
```
female_analysis_extensive <- bmbstats::describe_data(x = height_data$Female,
  estimator_function = bmbstats::data_estimators, control = model_control(seed = 1667,
  boot_type = "bca", boot_samples = 2000, confidence = 0.95))

female_analysis_extensive
#> Bootstrap with 2000 resamples and 95% bca confidence intervals.
#>
```

13.2. VISUALIZATION AND ANALYSIS OF A SINGLE GROUP/VARIABLE 203

```
#>   estimator      value      lower      upper
#>   mean 163.177916 160.913435 165.332557
#>   SD    8.199373  7.017929  9.874733
#>   CV % 5.024806  4.293769  6.039070
#>   median 164.003882 158.851885 165.294917
#>   mode 164.940229 156.333306 170.084573
#>   MAD   8.857033  6.340600 12.163640
#>   IQR   11.669487  8.447784 16.327453
#>   min   145.593392        NA        NA
#>   max   181.121685 175.163488 181.121685
#>   range 35.528292  28.764563  35.528292
```

```
plot(female_analysis_extensive)
```



As can be seen from the bootstrap estimators distribution, some estimators, like `mode`, `median`, `min`, `max` and `range` have *weird* distribution and their CIs should not be trusted.

Robust estimators is a family of estimators that are *robust* to outliers. Here I will use `median` and 10 and 20% *trimmed mean*. Trimming involves removing certain percentage of top and bottom observations from the sample, which removes potential outliers.

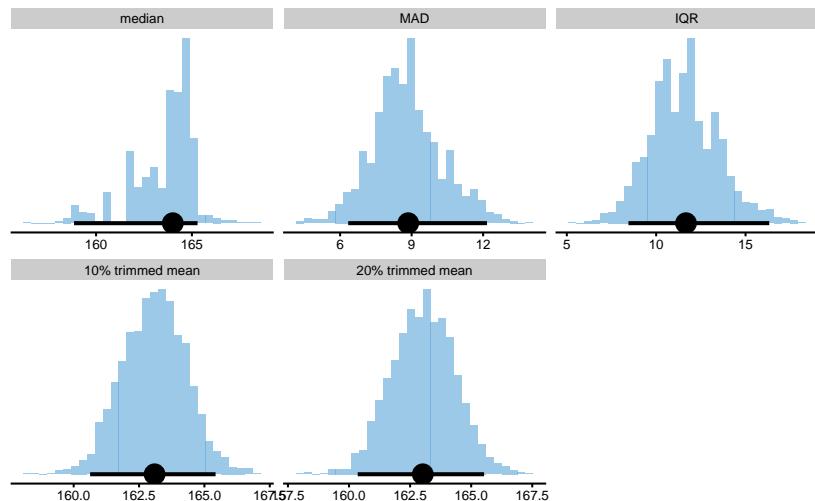
```
female_analysis_robust <- bmbstats::describe_data(x = height_data$Female,
estimator_function = bmbstats::data_estimators_robust,
control = model_control(seed = 1667, boot_type = "bca",
boot_samples = 2000, confidence = 0.95))
```

```

female_analysis_robust
#> Bootstrap with 2000 resamples and 95% bca confidence intervals.
#>
#>      estimator      value     lower      upper
#>      median 164.003882 158.851885 165.29492
#>      MAD    8.857033   6.340600 12.16364
#>      IQR   11.669487   8.447784 16.32745
#> 10% trimmed mean 163.095407 160.630145 165.42765
#> 20% trimmed mean 163.020558 160.356299 165.52687

plot(female_analysis_robust)

```



The simplicity of the bootstrap is that it can provide CIs for any estimator you can think of. But as always, Devil is in the details and some CIs for certain estimators (or small sample sizes) cannot be trusted and can be biased. This topic is beyond this book. The easiest test you can do is to run a simulation and see if the Type I error rates are not inflated.

To summarize the analysis of the single sample, let's say that I am interested in estimating proportion of the females taller than 180cm in the population using the sample acquired. I can easily write my own estimator function and use normal or t-distribution to estimate the proportion, then plug that into the bootstrap to get CIs:

```

# Estimators function
prop_estimators <- function(x, na.rm = FALSE) {
  mean_x <- mean(x, na.rm = na.rm)

```

```

sd_x <- sd(x, na.rm = na.rm)
n_x <- length(x)

# Use t-distribution to calculate proportion over 180cm
prop_t <- 1 - pt((180 - mean_x)/sd_x, df = n_x - 1)

# Use normal distribution to calculate proportion over
# 180cm
prop_norm <- 1 - pnorm(180, mean = mean_x, sd = sd_x)

# Use `brute-force` (simple counting) to calculate
# proportion over 180cm
prop_brute <- sum(x > 180)/n_x

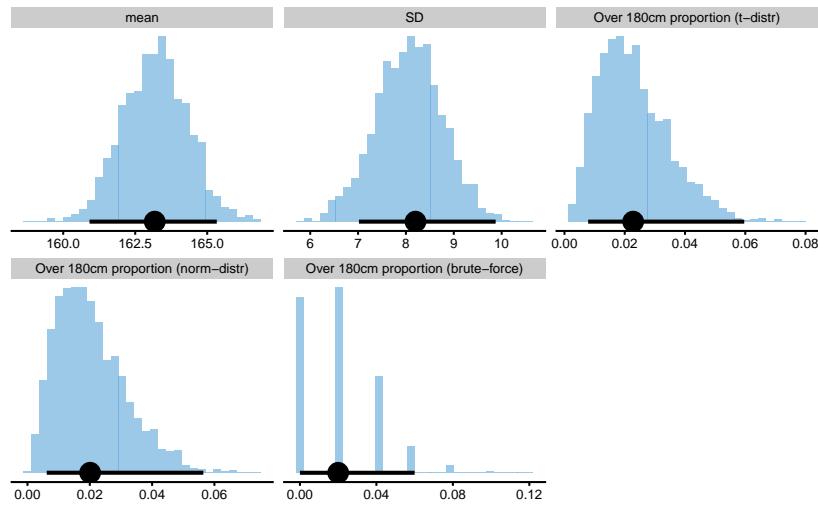
return(c(mean = mean_x, SD = sd_x, `Over 180cm proportion (t-distr)` = prop_t,
        `Over 180cm proportion (norm-distr)` = prop_norm,
        `Over 180cm proportion (brute-force)` = prop_brute))
}

tall_females <- bmbstats::describe_data(x = height_data$Female,
                                         estimator_function = prop_estimators, control = model_control(seed = 1667,
                                         boot_type = "bca", boot_samples = 2000, confidence = 0.95))

tall_females
#> Bootstrap with 2000 resamples and 95% bca confidence intervals.
#>
#>           estimator      value
#>             mean 163.17791620
#>             SD   8.19937316
#> Over 180cm proportion (t-distr) 0.02278501
#> Over 180cm proportion (norm-distr) 0.02010279
#> Over 180cm proportion (brute-force) 0.02000000
#>       lower      upper
#> 1.609134e+02 165.33255732
#> 7.017929e+00  9.87473324
#> 7.860132e-03  0.05965420
#> 6.171549e-03  0.05644849
#> 0.000000e+00  0.06000000

plot(tall_females)

```



13.3 Visualization and analysis of the two independent groups

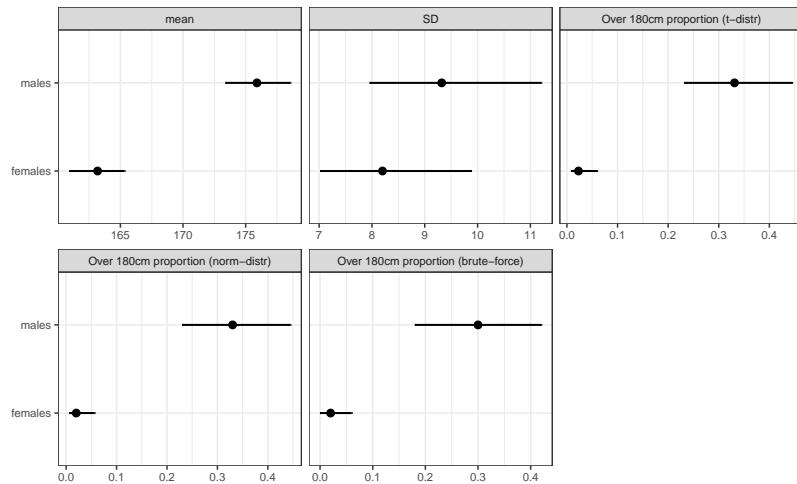
Estimators for each independent group (e.g. males and females) can be visualized side by side. For example, we might be interested in visualizing `mean`, `SD` and proportion over 180cm for males vs. females:

```
tall_males <- bmbstats::describe_data(x = height_data$Male,
estimator_function = prop_estimators, control = model_control(seed = 1667,
boot_type = "bca", boot_samples = 2000, confidence = 0.95))

compare_groups <- rbind(data.frame(group = "females", tall_females$estimators),
data.frame(group = "males", tall_males$estimators))

ggplot(compare_groups, aes(y = group, x = value)) + theme_bw(8) +
geom_errorbarh(aes(xmax = upper, xmin = lower), color = "black",
height = 0) + geom_point() + xlab("") + ylab("") +
facet_wrap(~estimator, scales = "free_x")
```

13.3. VISUALIZATION AND ANALYSIS OF THE TWO INDEPENDENT GROUPS 207



As can be seen from the figure, males have higher `mean` height, higher `SD` (but not sure if it is *statistically significant* nor *practically significant* - you can check this later with a few `bmbstats` functions) and higher proportion of individual over 180cm. Rather than comparing individual group estimates, we can perform independent group analysis using `bmbstats::compare_independent_groups`. But before we do that, let's plot the groups using `bmbstats::plot_raincloud` function. To do that, we need to convert our *wide* height data to *long* format:

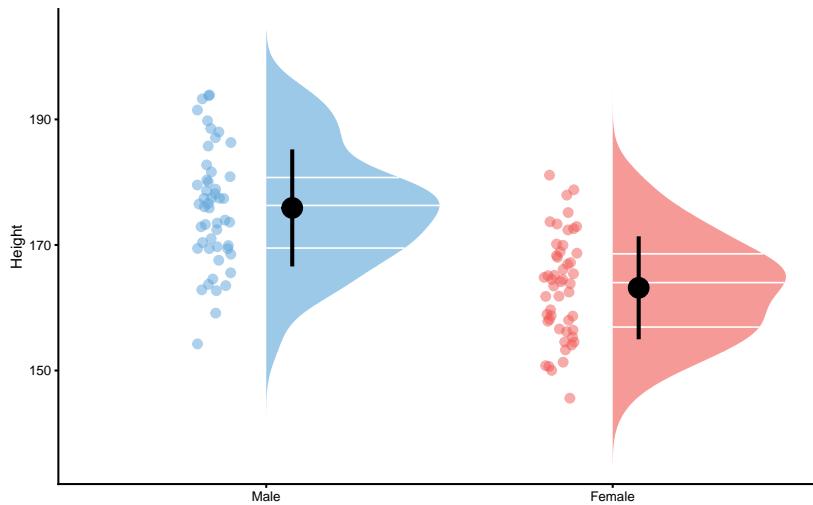
```
height_data_long <- gather(height_data, key = "Gender", value = "Height")

# Order factors
height_data_long$Gender <- factor(height_data_long$Gender,
                                     levels = c("Male", "Female"))

head(height_data_long)
#>   Gender   Height
#> 1  Male 193.9007
#> 2  Male 172.4291
#> 3  Male 186.3210
#> 4  Male 177.4417
#> 5  Male 167.5636
#> 6  Male 172.9078
```

And now we can plot the group height distribution:

```
bmbstats::plot_raincloud(data = height_data_long, value = "Height",
                           group = "Gender")
```



To perform descriptive analysis of the independent groups, we will use `bmbstats::compare_independent_groups` function. This function use estimator function `bmbstats::independent_groups_estimators` that provide all the major estimators introduced in the [Comparing dependent groups](#) section. For the SESOI we will use 2.5cm, like we have done in the [Comparing dependent groups](#) section as well:

```
independent_groups_estimators(group_a = height_data$Female,
                             group_b = height_data$Male, SESOI_lower = -2.5, SESOI_upper = 2.5)
#>      SESOI lower      SESOI upper
#>      -2.50000000      2.50000000
#>      SESOI range      Mean diff
#>      5.00000000     12.72521903
#>      SD diff          SD pooled
#>      12.41402468     8.77804103
#>      %CV diff         % diff
#>      97.55450689     7.79837084
#>      Ratio            Cohen's d
#>      1.07798371     1.44966502
#>      CLES              OVL
#>      0.84733444     0.46855479
#>      Mean diff to SESOI  SD diff to SESOI
#>      2.54504381     2.48280494
#>      pLower            pEquivalent
#>      0.11148343     0.09457649
#>      pHiger           0.79394008
```

`bmbstats::compare_independent_groups` uses by default the `bmbstats::independent_groups_estimators`

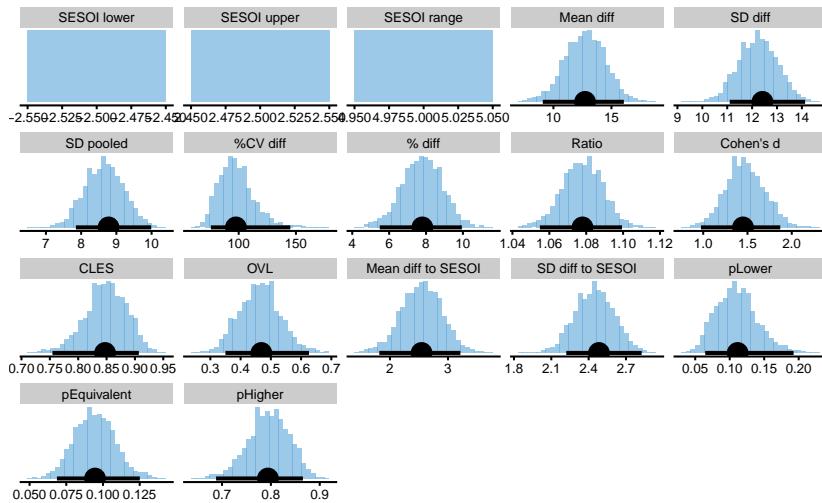
13.3. VISUALIZATION AND ANALYSIS OF THE TWO INDEPENDENT GROUPS 209

but we can write our own estimators function a bit later:

```
males_females_comp <- compare_independent_groups(group_a = height_data$Female,
                                                 group_b = height_data$Male, SESOI_lower = -2.5, SESOI_upper = 2.5)
#> [1] "All values of t are equal to 2.5 \n Cannot calculate confidence intervals"
#> [1] "All values of t are equal to 5 \n Cannot calculate confidence intervals"

males_females_comp
#> Bootstrap with 2000 resamples and 95% bca confidence intervals.
#>
#>      estimator      value     lower      upper
#>      SESOI lower -2.50000000      NA      NA
#>      SESOI upper  2.50000000      NA      NA
#>      SESOI range  5.00000000      NA      NA
#>      Mean diff  12.72521903  9.08380753 16.0686119
#>      SD diff   12.41402468 11.10081268 14.1287776
#>      SD pooled  8.77804103  7.84945992  9.9905544
#>      %CV diff  97.55450689 75.67596790 145.1388909
#>      % diff    7.79837084  5.48856008  9.9411055
#>      Ratio     1.07798371  1.05488560  1.0994111
#>      Cohen's d  1.44966502  0.97452326  1.8701976
#>      CLES       0.84733444  0.75460141  0.9069431
#>      OVL        0.46855479  0.34974837  0.6260860
#>      Mean diff to SESOI 2.54504381  1.81676151  3.2137224
#>      SD diff to SESOI 2.48280494  2.22016254  2.8257555
#>      pLower     0.11148343  0.06449905  0.1926671
#>      pEquivalent 0.09457649  0.06862640  0.1250760
#>      pHIGHER    0.79394008  0.68781913  0.8655902

plot(males_females_comp)
```



You can notice that SESOI threshold doesn't have any bootstrap distribution. That is because we have provide *a priori* SESOI. We can also estimate SESOI within the bootstrap loop. For SESOI we can use pooled SD of the `group_a` and `group_b` multiplied by 0.2, which represents Cohen's trivial magnitude. This is the default behavior of the `bmbstats::compare_independent_groups` and other similar functions. You can write your own function for estimating SESOI by providing function argument to `SESOI_lower` and `SESOI_upper` parameters. For the sake of example, I will do that here, but only for the `SESOI_upper` and will stick to -2.5cm for the `SESOI_lower`:

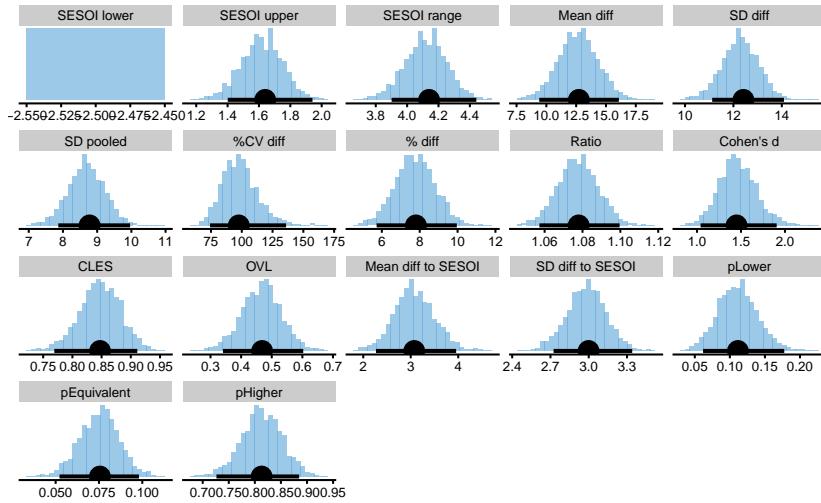
```
males_females_comp_est <- compare_independent_groups(group_a = height_data$Female,
  group_b = height_data$Male, SESOI_lower = -2.5, SESOI_upper = function(group_a,
    group_b, na.rm) {
    sd(group_a, na.rm = na.rm) * 0.2
  })

males_females_comp_est
#> Bootstrap with 2000 resamples and 95% bca confidence intervals.
#>
#>      estimator      value      lower
#>      SESOI lower -2.500000000      NA
#>      SESOI upper  1.63987463  1.40101324
#>      SESOI range  4.13987463  3.90101324
#>      Mean diff 12.72521903  9.39979860
#>      SD diff 12.41402468 11.12171677
#>      SD pooled 8.77804103  7.86424134
#>      %CV diff 97.55450689 74.22144558
#>      % diff  7.79837084  5.72302737
```

13.3. VISUALIZATION AND ANALYSIS OF THE TWO INDEPENDENT GROUPS 211

```
#>           Ratio  1.07798371  1.05723027
#>           Cohen's d  1.44966502  1.04057626
#>           CLES   0.84733444  0.76901630
#>           DVL    0.46855479  0.34069164
#> Mean diff to SESOI  3.07381749  2.26756237
#> SD diff to SESOI  2.99864749  2.72442801
#> pLower   0.11148343  0.06178746
#> pEquivalent  0.07554712  0.05221579
#> pHIGHER   0.81296945  0.72639255
#> upper
#> NA
#> 1.94141198
#> 4.44141198
#> 16.07945845
#> 14.08901524
#> 9.96243822
#> 135.97174823
#> 9.95978160
#> 1.09959782
#> 1.90567697
#> 0.91107613
#> 0.60291087
#> 3.95663372
#> 3.34078108
#> 0.17758243
#> 0.09808881
#> 0.88497505
```

```
plot(males_females_comp_est)
```



You can now notice that SESOI upper and SESOI range have bootstrap distribution. It is important that if we estimate SESOI from the obtained sample, the SESOI estimation *must* be in the bootstrap loop, and our uncertainty about it's estimate must be propagated to other estimators that uses SESOI (e.g. pLower, pEquivalent, pHiger). To demonstrate the difference, consider the following two analyses using different SESOI estimation. One estimates SESOI *inside* the bootstrap loop, and the other estimates SESOI *outside* the bootstrap loop. To make sure the same bootstrap is performed, we will set the same seed parameter:

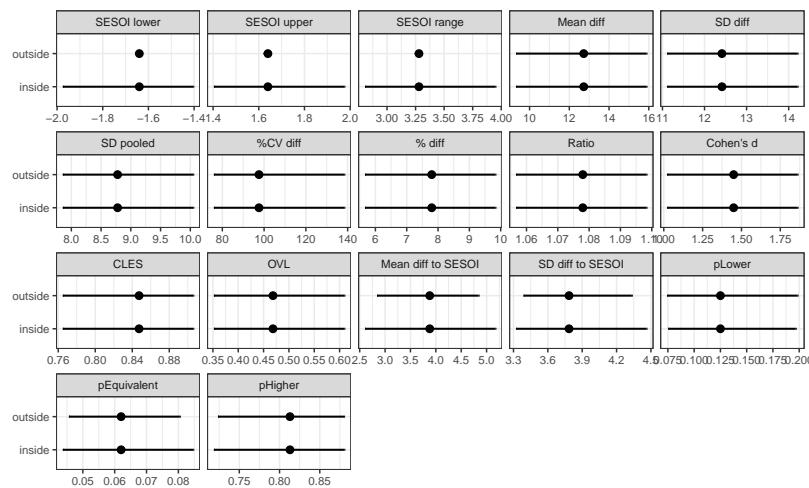
```
# SESOI estimated inside the bootstrap loop
males_females_comp_inside <- compare_independent_groups(group_a = height_data$Female,
  group_b = height_data$Male, SESOI_lower = function(group_a,
    group_b, na.rm) {
    -sd(group_a, na.rm = na.rm) * 0.2
  }, SESOI_upper = function(group_a, group_b, na.rm) {
    sd(group_a, na.rm = na.rm) * 0.2
  }, control = model_control(seed = 1667))

# SESOI estimated outside the bootstrap loop
males_females_comp_outside <- compare_independent_groups(group_a = height_data$Female,
  group_b = height_data$Male, SESOI_lower = -sd(height_data$Female) *
  0.2, SESOI_upper = sd(height_data$Female) * 0.2,
  control = model_control(seed = 1667))
#> [1] "All values of t are equal to 1.63987463256618 \n Cannot calculate confidence"
#> [1] "All values of t are equal to 3.27974926513235 \n Cannot calculate confidence"

# Plot the estimators
compare_analyses <- rbind(data.frame(group = "inside", males_females_comp_inside$estima
```

```
data.frame(group = "outside", males_females_comp_outside$estimators))

ggplot(compare_analyses, aes(y = group, x = value)) + theme_bw(8) +
  geom_errorbarh(aes(xmax = upper, xmin = lower), color = "black",
                 height = 0) + geom_point() + xlab("") + ylab("") +
  facet_wrap(~estimator, scales = "free_x")
```



The difference in CIs for the estimators that uses SESOI (i.e. magnitude-based estimators) is not staggering, but CIs are wider for the *inside* method (for example, compare **Mean diff** to **SESOI estimator**).

13.4 NHST, METs and MBI functions

Is there are statistically significant difference in the **mean** height between males and females? As explained in the **Statistical inference** and **Bootstrap** chapters, we can derive **p-value** using bootstrap resamples. Let's test the Null Hypothesis that **mean** difference between males and females is 0. To do that we will used **bmbstats::bootstrap_NHST** function that demands us to select estimator of interest using **estimator** parameter. **bmbstats::bootstrap_NHST** uses the result object from the **bmbstats::compare_independent_groups** function:

```
males_females_NHST <- bmbstats::bootstrap_NHST(males_females_comp,
                                                 estimator = "Mean diff", null_hypothesis = 0, test = "two.sided")

males_females_NHST
#> Null-hypothesis significance test for the `Mean diff` estimator
```

```
#> Bootstrap result: Mean diff=12.725, 95% CI [9.084, 16.069]
#> H0=0, test: two.sided
#> p=0.000499750124937531
```

As can be seen, the **p-value** is below 0.05 (i.e. *alpha*), so we can conclude that the **mean** difference is statistically significant.

What if our hypothesis is that males are taller by females by at least 10cm? We can also test that hypothesis using *one-sided* NHST:

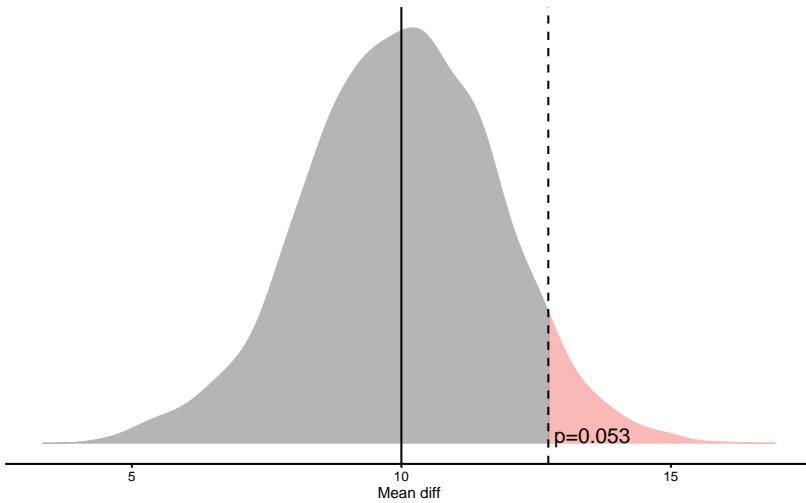
```
males_females_NHST <- bmbstats::bootstrap_NHST(males_females_comp,
    estimator = "Mean diff", null_hypothesis = 10, test = "greater")

males_females_NHST
#> Null-hypothesis significance test for the `Mean diff` estimator
#> Bootstrap result: Mean diff=12.725, 95% CI [9.084, 16.069]
#> H0=10, test: greater
#> p=0.0535
```

Estimated **p-value** is slightly over 0.05, so we do not reject the Null Hypothesis. This is interesting result, particularly since we know that the *true mean* difference is around 12.7cm (true **mean** height for males is 177.8cm and for females is 165.1cm). It could be that this sample size is under-powered to detect this small difference between Null Hypothesis (i.e. $>10\text{cm}$) and Alternative Hypothesis - true difference in this case - of 12.7cm. This topic is discussed in the [Statistical Power](#). This could easily be tested with a simulation, although that is beyond the scope of this chapter.

Graphically, this test looks like this:

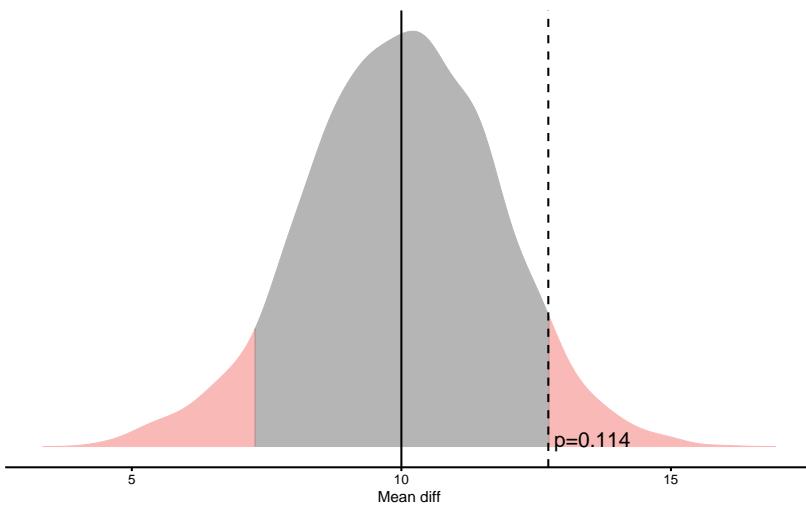
```
plot(males_females_NHST)
```



In the above graph, the estimator bootstrap distribution (i.e. `mean difference`) is centered around Null Hypothesis (i.e. 10cm). If we perform this same test, but using two sided NHST (which is default), we will get the following result and plot:

```
males_females_NHST <- bmbstats::bootstrap_NHST(males_females_comp,
  estimator = "Mean diff", null_hypothesis = 10)

plot(males_females_NHST)
```



We have already decided that our SESOI in height is 2.5cm. But this SESOI is related to individual observations, not necessarily to estimators (i.e. `mean`

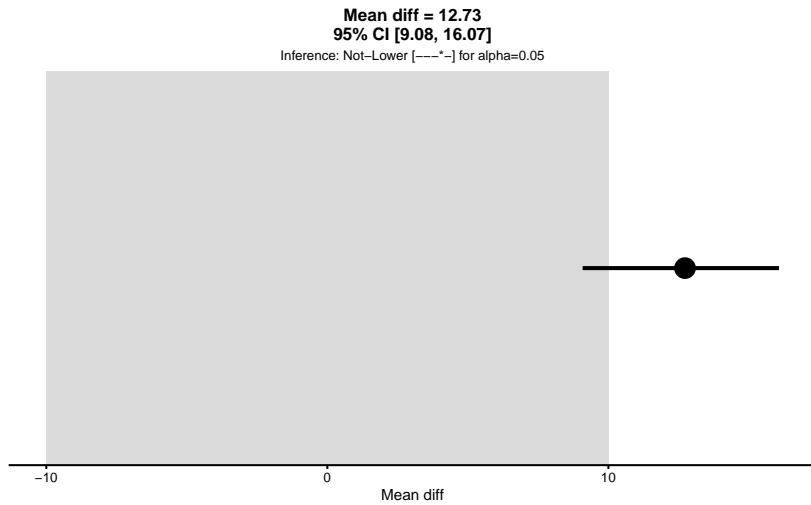
difference). This topic is discussed in the Individual vs. Parameter SESOI section of this book. Let's use 10cm as SESOI for the mean difference estimator and perform METs using `bmbstats::bootstrap_MET` function and alpha level set to 0.05:

```
males_females_MET <- bmbstats::bootstrap_MET(males_females_comp,
  estimator = "Mean diff", SESOI_lower = -10, SESOI_upper = 10,
  alpha = 0.05)

males_females_MET
#> Minimum effect tests for the `Mean diff` estimator
#> Bootstrap result: Mean diff=12.725, 95% CI [9.084, 16.069]
#> SESOI: [-10, 10], alpha=0.05
#>
#>           Test      p.value
#>   inferiority 1.00000000000
#> non-superiority 0.94650000000
#> equivalence 0.94650000000
#> non-inferiority 0.0004997501
#> superiority 0.05350000000
#>
#> Final inference: Not-Lower
```

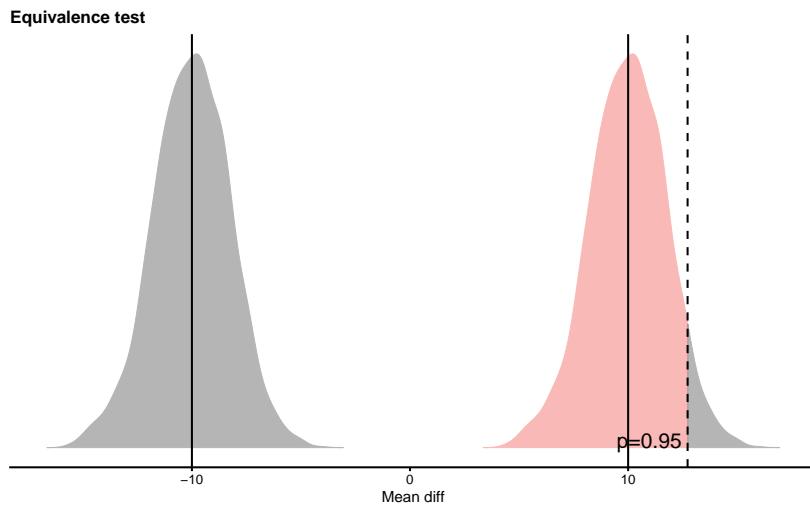
As can be seen from the result of the MET analysis, the final inference is that male height is “not lower” than female height using SESOI of 10cm. I personally prefer this to be conveyed visually by using `plot` function:

```
plot(males_females_MET)
```

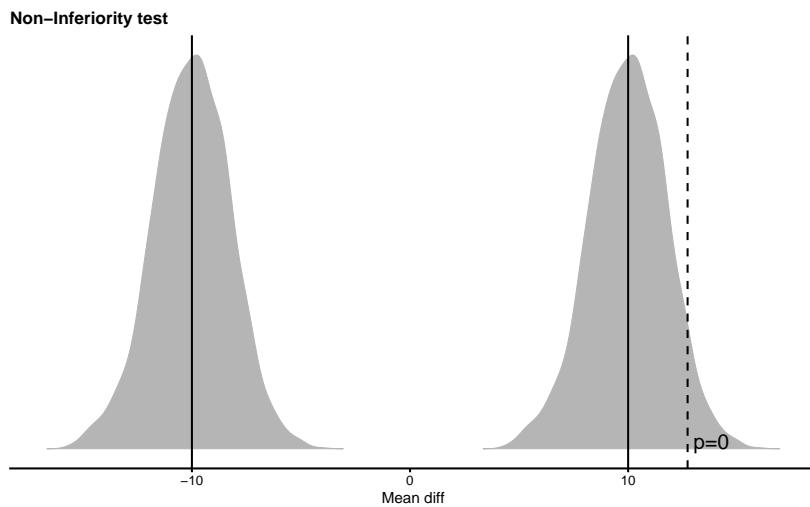


We can also plot each individual MET test, for example *equivalence* and *non-inferiority* tests:

```
plot(males_females_MET, type = "equivalence")
```



```
plot(males_females_MET, type = "non-inferiority")
```



What if I decide about different SESOI values or different Null Hypothesis or even alpha levels? Well, that is an example of *p-hacking* discussed in the [Statistical Power](#) section. P-hacking represents *hypothesizing after results are known*, or in

other words tuning your analysis to be more acceptable for publications. That's why it is important for the *confirmatory* studies to have all the threshold and the analysis *a priori* defined or pre-registered.

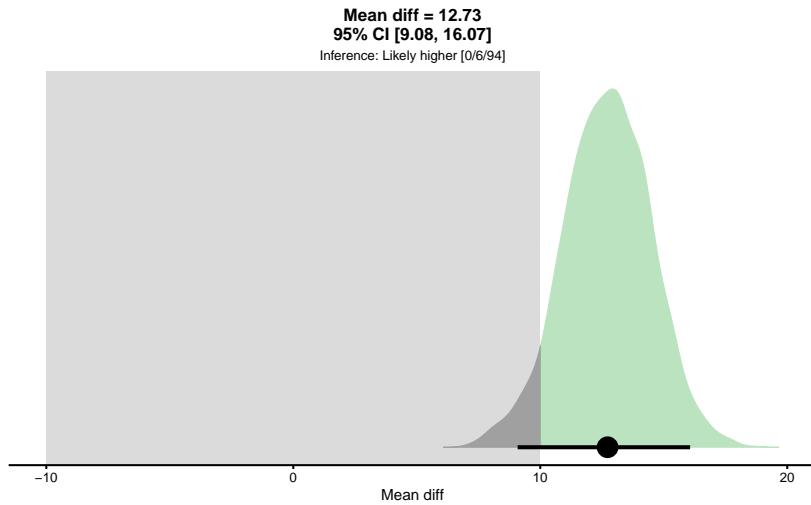
To perform MBI, use `bmbstats::bootstrap_MBI` function:

```
males_females_MBI <- bmbstats::bootstrap_MBI(males_females_comp,
  estimator = "Mean diff", SESOI_lower = -10, SESOI_upper = 10)

males_females_MBI
#> Magnitude-based inference for the `Mean diff` estimator
#> Bootstrap result: Mean diff=12.725, 95% CI [9.084, 16.069]
#> SESOI: [-10, 10]
#>
#>      Test prob
#>      lower 0.00
#> equivalent 0.06
#>      higher 0.94
#>
#> Final inference: Likely higher
```

The final inference of the MBI is “likely higher” mean difference. As always, plotting is much more informational:

```
plot(males_females_MBI)
```



13.5 Comparing two dependent groups

13.5.1 Measurement error issues

In the [Description](#) chapter, to showcase comparison between two dependent groups, bench press data involving Pre-test and Post-test observations were used. Let's create that data-set here. To demonstrate measurement error effects, on top of *true* Pre- and Post- 1RM test scores, I will add measurement error that is normally distributed with SD equal to 2.5kg. In the first example, there will be no *systematic* nor *random* change between Pre-test and Post-test, thus measurement error will be solely responsible for the observed change (although there will be no *true* change).

```
set.seed(1666)

n_subjects <- 20

measurement_error <- 2.5

systematic_change <- 0
random_change <- 0

bench_press_data <- tibble(
  # Generate athlete name
  Athlete = factor(paste(
    "Athlete",
    str_pad(
      string = seq(1, n_subjects),
      width = 2,
      pad = "0"
    )
  )),
  # True Pre-test
  `Pre-test (true)` = rnorm(
    n = n_subjects,
    mean = 100,
    sd = 7.5
  ),
  # True Change
  `Change (true)` = rnorm(
    n = n_subjects,
    mean = systematic_change,
    sd = random_change
  )
)
```

```

),
# True Post-test
`Post-test (true)` = `Pre-test (true)` + `Change (true)`,

# Observed Pre-test
`Pre-test (observed)` = `Pre-test (true)` +
# Add measurement error
rnorm(
  n = n_subjects,
  mean = 0,
  sd = measurement_error
),

# Observed Post-test
`Post-test (observed)` = `Post-test (true)` +
# Add measurement error
rnorm(
  n = n_subjects,
  mean = 0,
  sd = measurement_error
),

# Observed Change
`Change (observed)` = `Post-test (observed)` - `Pre-test (observed)`
)

bench_press_data
#> # A tibble: 20 x 7
#>   Athlete `Pre-test (true)` `Change (true)`
#>   <fct>      <dbl>        <dbl>
#> 1 Athlet~    111.         0
#> 2 Athlet~    102.         0
#> 3 Athlet~    93.4         0
#> 4 Athlet~    95.4         0
#> 5 Athlet~    111.         0
#> 6 Athlet~    110.         0
#> 7 Athlet~    104.         0
#> 8 Athlet~    93.7         0
#> 9 Athlet~    99.6         0
#> 10 Athlet~   106.         0
#> 11 Athlet~   102.         0
#> 12 Athlet~   101.         0
#> 13 Athlet~   92.9         0
#> 14 Athlet~   98.2         0

```

```
#> 15 Athlet~     88.3      0
#> 16 Athlet~    106.       0
#> 17 Athlet~    95.8      0
#> 18 Athlet~    92.9      0
#> 19 Athlet~    103.       0
#> 20 Athlet~    104.       0
#> # ... with 4 more variables: `Post-test (true)` <dbl>,
#> #   `Pre-test (observed)` <dbl>, `Post-test
#> #   (observed)` <dbl>, `Change (observed)` <dbl>
```

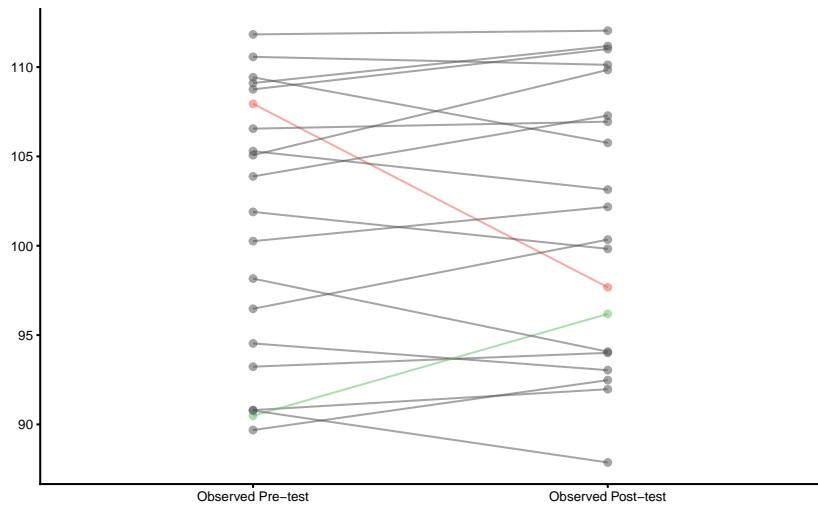
Let's plot the true Pre-test and Post-test scores using a scatter plot and SESOI band of -5 to 5kg:

```
plot_pair_changes(group_a = bench_press_data$`Pre-test (true)` ,
                  group_b = bench_press_data$`Post-test (true)` , group_a_label = "True Pre-test",
                  group_b_label = "True Post-test", SESOI_lower = -5, SESOI_upper = 5)
```



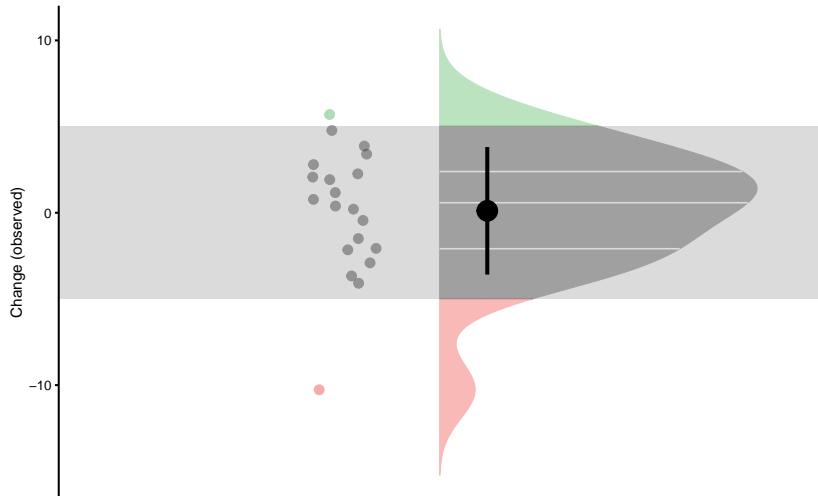
As can be seen, there is no true change. Let's see what happens when we plot observed scores:

```
plot_pair_changes(group_a = bench_press_data$`Pre-test (observed)` ,
                  group_b = bench_press_data$`Post-test (observed)` , group_a_label = "Observed Pre-test",
                  group_b_label = "Observed Post-test", SESOI_lower = -5, SESOI_upper = 5)
```



We can also plot distribution of the Change scores:

```
plot_raincloud_SESOI(bench_press_data, value = "Change (observed)",
                      SESOI_lower = -5, SESOI_upper = 5)
```



We would be very quick to claim that there are individuals that demonstrated higher or lower change (compared to SESOI). But remember that in this data set there is not *true* change - implying that all observed change is due to measurement error.

Since we know that there is no true change, let's do the summary of the observed Change score:

```
mean(bench_press_data$`Change (observed)`)  
#> [1] 0.1114846  
  
sd(bench_press_data$`Change (observed)`)  
#> [1] 3.699529
```

Since we know that the measurement error is 2.5kg, the SD of the change score is expected to be $2.5 \times \sqrt{2}$ or 3.54kg.

Using `bmbstats::describe_data` we can perform bootstrap CIs for the `mean` and SD of the Change score:

```
obs_change_analysis <- bmbstats::describe_data(x = bench_press_data$`Change (observed)`,  
                                              estimator_function = bmbstats::data_estimators_simple,  
                                              control = model_control(seed = 1667, boot_type = "perc",  
                                              boot_samples = 1000, confidence = 0.9))  
  
obs_change_analysis  
#> Bootstrap with 1000 resamples and 90% perc confidence intervals.  
#>  
#>   estimator      value    lower     upper  
#>   mean 0.1114846 -1.397619 1.360018  
#>   SD   3.6995292  2.405751 4.775318
```

The 90% CIs for the SD of the observed Change score captures expected change measurement error of 3.54kg. We will come back to this in the [Reliability](#) section.

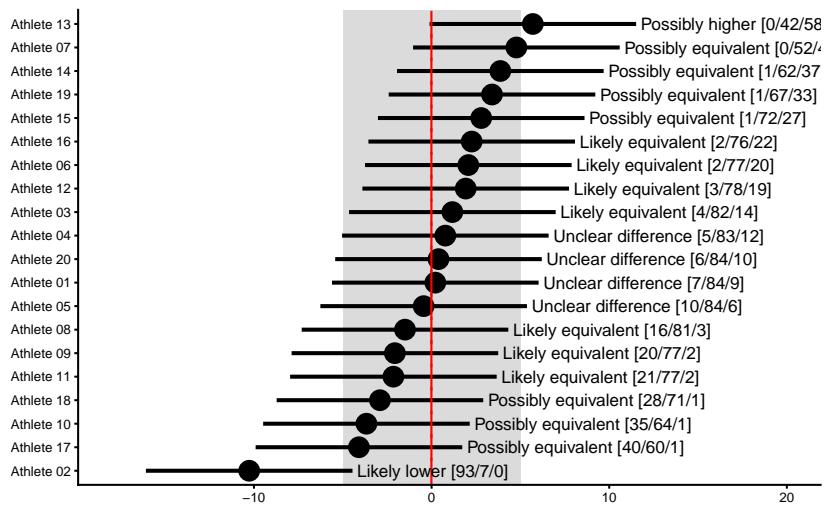
Since we are aware of the measurement error involved in our observations, we can perform MBI and MET of the observed change scores. MBI can be performed using `bmbstats::observations_MBI` function. If we plot the result, confidence intervals (error bars) represent SDC (smallest detectable change), which is measurement error multiplied with appropriate critical value to get desired confidence level. On top of plotting MBI, I will also plot the true score using `true_observations` of the `plot` function (indicated by red line):

```
obs_change_MBI <- bmbstats::observations_MBI(  
  observations = bench_press_data$`Change (observed)`,  
  observations_label = bench_press_data$Athlete,  
  measurement_error = 2.5 * sqrt(2),  
  # Degrees of freedom from the reliability study. Use `Inf` for normal distribution  
  df = Inf,  
  SESOI_lower = -5,  
  SESOI_upper = 5,  
  confidence = 0.9  
)
```

```

plot(
  obs_change_MBI,
  true_observations = bench_press_data$`Change (true)` ,
  control = plot_control(points_size = 5)
) +
  xlim(-18, 20)

```



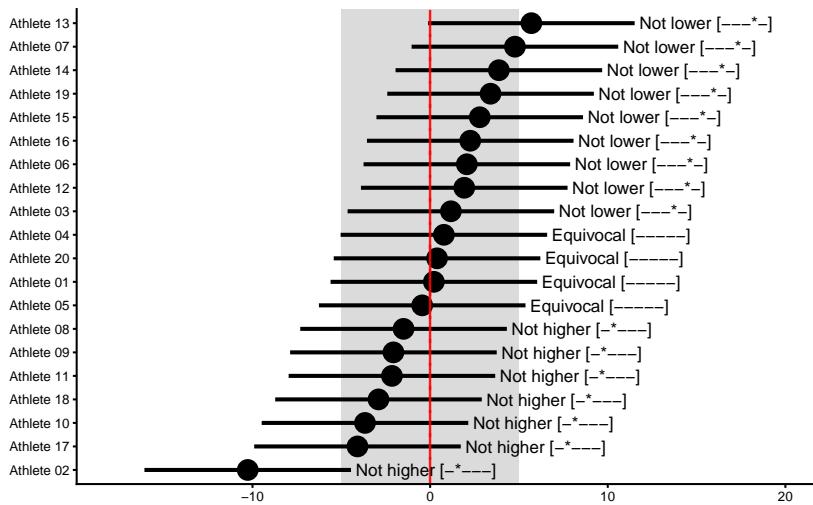
To perform METs, use `bmbstats::observations_MET` function:

```

obs_change_MET <- bmbstats::observations_MET(
  observations = bench_press_data$`Change (observed)` ,
  observations_label = bench_press_data$Athlete,
  measurement_error = 2.5 * sqrt(2),
  # Degrees of freedom from the reliability study. Use `Inf` for normal distribution
  df = Inf,
  SESOI_lower = -5,
  SESOI_upper = 5,
  alpha = 0.05,
  confidence = 0.9
)

plot(
  obs_change_MET,
  true_observations = bench_press_data$`Change (true)` ,
  control = plot_control(points_size = 5)
) +
  xlim(-18, 20)

```



It seems like the Athlete 02 showed *true* Change, but since we generated the data we know that there is no *true* Change (also check the vertical red lines for the *true* Change scores). Thus, if we conclude that this individual showed lower change, we would be making *Type I* error. Since we are performing multiple individual tests, we could/should *adjust* the alpha parameter (e.g. by dividing it by number of tests, or in this case athletes - *Bonferroni correction*) to avoid *inflating* family-wise error rates, since particular athlete can show significant change due to chance alone (due to multiple comparisons/test). To do that, simply divide alpha by the number of athletes:

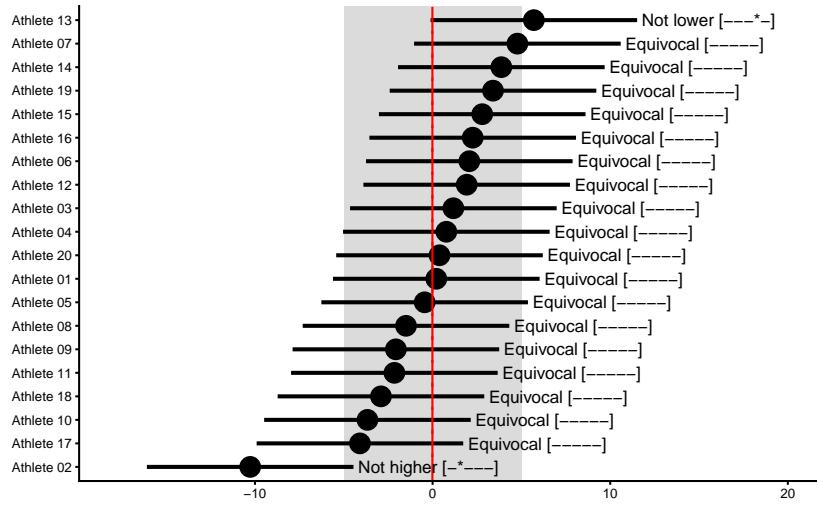
```

obs_change_MET <- bmbstats::observations_MET(
  observations = bench_press_data$`Change (observed)`,
  observations_label = bench_press_data$Athlete,
  measurement_error = 2.5 * sqrt(2),
  # Degrees of freedom from the reliability study. Use `Inf` for normal distribution
  df = Inf,
  SESOI_lower = -5,
  SESOI_upper = 5,
  alpha = 0.05 / n_subjects,

  # Confidence could be adjusted as well
  # but it is used mainly for plotting
  confidence = 0.9
)

plot(
  obs_change_MET,
  true_observations = bench_press_data$`Change (true)`,
```

```
control = plot_control(points_size = 5)
) +
  xlim(-18, 20)
```



The point of this analysis is that we need to know measurement error to infer about true change in individuals. Since we do know that there is no real change in this example, we can see how measurement error cause wrong inferences about the true changes.

Let's now generate the data with true changes, where systematic change is 10kg and random change is 10kg as well:

```
set.seed(1666)

n_subjects <- 20

measurement_error <- 2.5

systematic_change <- 10
random_change <- 10

bench_press_data <- tibble(
  # Generate athlete name
  Athlete = factor(paste(
    "Athlete",
    str_pad(
      string = seq(1, n_subjects),
      width = 2,
```

```
    pad = "0"
)
)),
# True Pre-test
`Pre-test (true)` = rnorm(
  n = n_subjects,
  mean = 100,
  sd = 7.5
),
# True Change
`Change (true)` = rnorm(
  n = n_subjects,
  mean = systematic_change,
  sd = random_change
),
# True Post-test
`Post-test (true)` = `Pre-test (true)` + `Change (true)`,

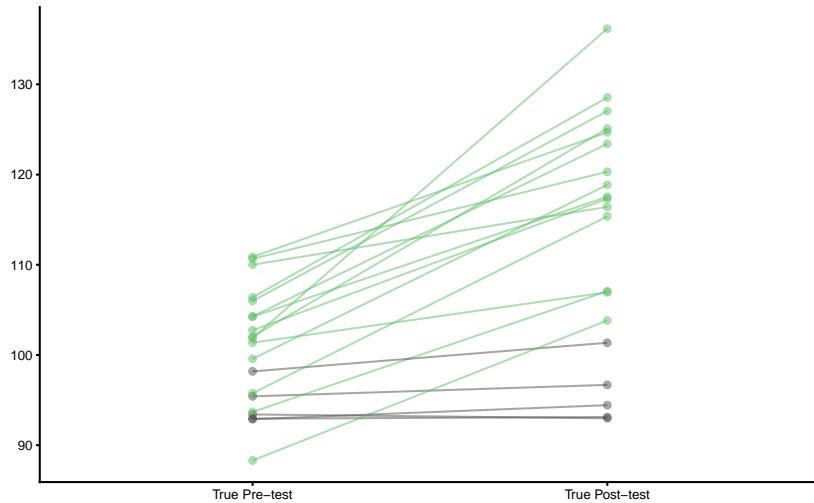
# Observed Pre-test
`Pre-test (observed)` = `Pre-test (true)` +
# Add measurement error
rnorm(
  n = n_subjects,
  mean = 0,
  sd = measurement_error
),
# Observed Post-test
`Post-test (observed)` = `Post-test (true)` +
# Add measurement error
rnorm(
  n = n_subjects,
  mean = 0,
  sd = measurement_error
),
# Observed Change
`Change (observed)` = `Post-test (observed)` - `Pre-test (observed)`
)

bench_press_data
#> # A tibble: 20 x 7
```

```
#>   Athlete `Pre-test` (true) `Change` (true)
#>   <fct>          <dbl>          <dbl>
#> 1 Athlet~       111.        13.8
#> 2 Athlet~       102.        34.3
#> 3 Athlet~       93.4       -0.428
#> 4 Athlet~       95.4        1.27
#> 5 Athlet~       111.        9.65
#> 6 Athlet~       110.        6.41
#> 7 Athlet~       104.        13.3
#> 8 Athlet~       93.7        13.4
#> 9 Athlet~       99.6        19.3
#> 10 Athlet~      106.        22.1
#> 11 Athlet~      102.        23.1
#> 12 Athlet~      101.        5.59
#> 13 Athlet~      92.9        0.190
#> 14 Athlet~      98.2        3.17
#> 15 Athlet~      88.3        15.5
#> 16 Athlet~      106.        21.1
#> 17 Athlet~      95.8        19.6
#> 18 Athlet~      92.9        1.54
#> 19 Athlet~      103.        14.6
#> 20 Athlet~      104.        19.1
#> # ... with 4 more variables: `Post-test` (true) `<dbl>`,
#> #   `Pre-test` (observed) `<dbl>`, `Post-test` (observed) `<dbl>`,
#> #   `Change` (observed) `<dbl>`
```

Let's plot true scores:

```
plot_pair_changes(group_a = bench_press_data$`Pre-test` (true),
                  group_b = bench_press_data$`Post-test` (true), group_a_label = "True Pre-test",
                  group_b_label = "True Post-test", SESOI_lower = -5, SESOI_upper = 5)
```



Since there are true changes (systematic and random) in this DGP, estimating mean and SD for the true Change scores will give us the estimate of the DGP parameters:

```
mean(bench_press_data$`Change (true)`)  
#> [1] 12.82946
```

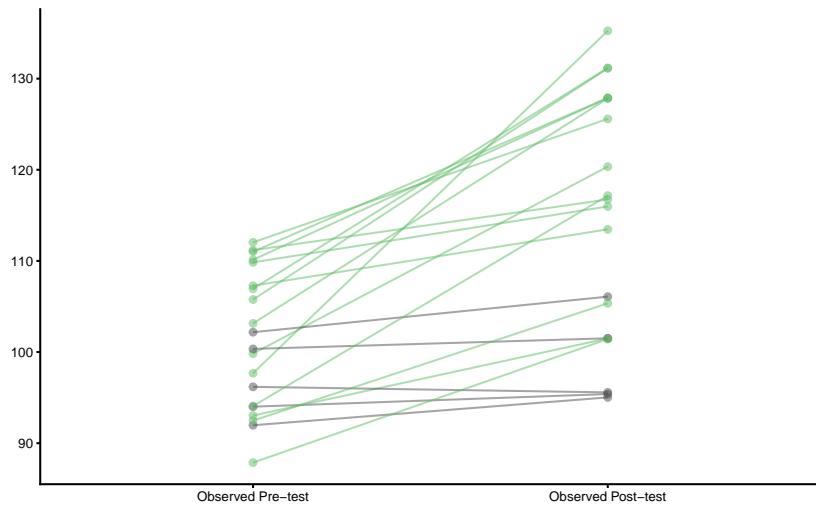
```
sd(bench_press_data$`Change (true)`)  
#> [1] 9.328114
```

To get bootstrap CI around these estimate, we can again use `bmbstats::describe_data` function:

```
true_change_analysis <- bmbstats::describe_data(x = bench_press_data$`Change (true)`,  
estimator_function = bmbstats::data_estimators_simple,  
control = model_control(seed = 1667, boot_type = "perc",  
boot_samples = 1000, confidence = 0.9))  
  
true_change_analysis  
#> Bootstrap with 1000 resamples and 90% perc confidence intervals.  
#>  
#>   estimator      value    lower     upper  
#>       mean 12.829463 9.402614 16.51982  
#>       SD   9.328114 6.781652 11.06663
```

The true DGP parameters (systematic effect of 10kg and random effect of 10kg) are captured with estimated CIs. Let's turn to observed scores:

```
plot_pair_changes(group_a = bench_press_data$`Pre-test (observed)`,
                  group_b = bench_press_data$`Post-test (observed)`, group_a_label = "Observed Pre-test",
                  group_b_label = "Observed Post-test", SESOI_lower = -5,
                  SESOI_upper = 5)
```



The image looks similar to true scores analysis. Let's estimate mean and SD CIs:

```
obs_change_analysis <- bmbstats::describe_data(x = bench_press_data$`Change (observed)`,
                                                estimator_function = bmbstats::data_estimators_simple,
                                                control = model_control(seed = 1667, boot_type = "perc",
                                                                           boot_samples = 1000, confidence = 0.9))

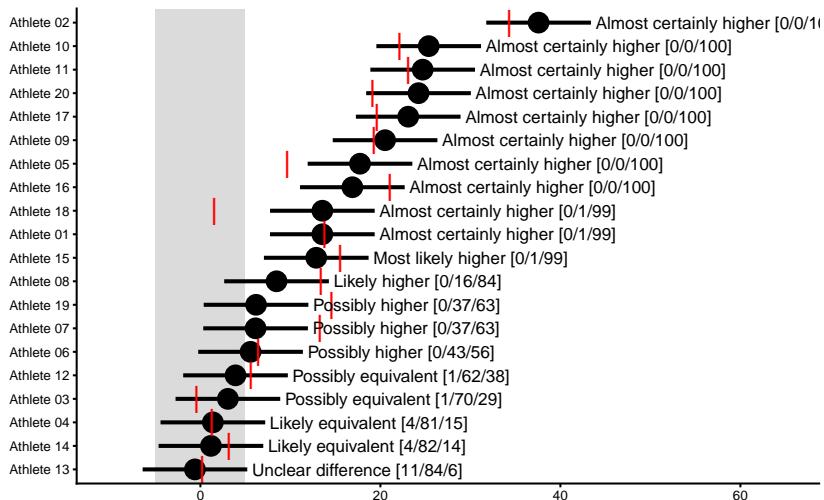
obs_change_analysis
#> Bootstrap with 1000 resamples and 90% perc confidence intervals.
#>
#>   estimator      value    lower     upper
#>   mean  13.26635 9.829395 17.26567
#>   SD    10.32696 7.533166 12.40584
```

As expected, the SD of the observed Change score is more inflated (than SD of the true Change score) due to measurement error. Before dealing with this issue, let's plot MBI and MET analysis results:

```
obs_change_MBI <- bmbstats::observations_MBI(
  observations = bench_press_data$`Change (observed)` ,
  observations_label = bench_press_data$Athlete,
  measurement_error = 2.5 * sqrt(2),
```

```
# Degrees of freedom from the reliability study. Use `Inf` for normal distribution
df = Inf,
SESOI_lower = -5,
SESOI_upper = 5,
confidence = 0.9
)

plot(
  obs_change_MBI,
  true_observations = bench_press_data$`Change (true)`,
  control = plot_control(points_size = 5)
) +
  xlim(-10, 65)
```



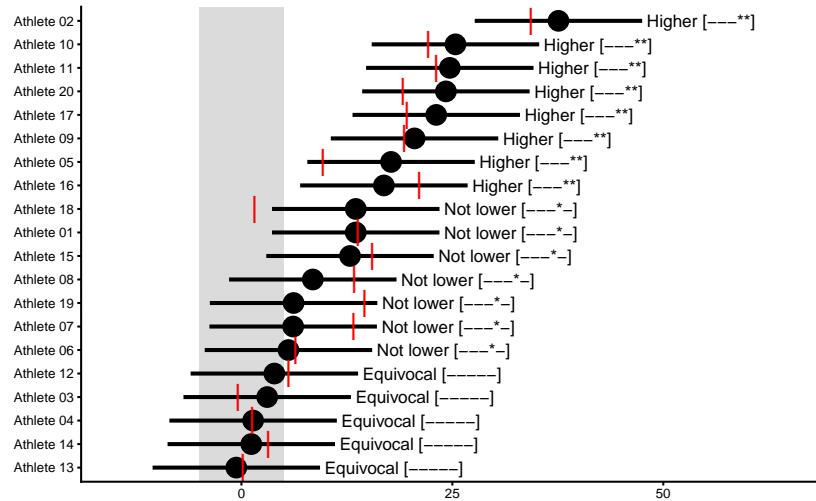
```
obs_change_MET <- bmbstats::observations_MET(
  observations = bench_press_data$`Change (observed)`,
  observations_label = bench_press_data$Athlete,
  measurement_error = 2.5 * sqrt(2),
  # Degrees of freedom from the reliability study. Use `Inf` for normal distribution
  df = Inf,
  SESOI_lower = -5,
  SESOI_upper = 5,
  alpha = 0.05 / n_subjects,
  # Will adjust CI for plotting as well
  confidence = 1 - (0.05 / n_subjects) * 2
)

plot(
```

```

obs_change_MET,
true_observations = bench_press_data$`Change (true)` ,
control = plot_control(points_size = 5)
) +
xlim(-15, 65)

```



Before jumping on the *responders vs. non-responders* bandwagon, it would be wise to check the statistical error committed by Dankel and Loenneke (Dankel and Loenneke 2019) pointed out in the letter-to-the-editor by Tenan *et al.* (Tenan, Vigotsky, and Caldwell, n.d.). This is a lesson to us all who are trying to come up with a *novel analyses*, like myself, so I am trying to be very cautious in using any bold statements.

13.5.2 Analysis of the dependent groups using `bmbstats::compare_dependent_`

To perform dependent group comparison, we will use `bmbstats::compare_dependent_groups` function, and `bmbstats::dependent_groups_estimators` estimator function. Let's first do it with the true Pre- and Post- scores:

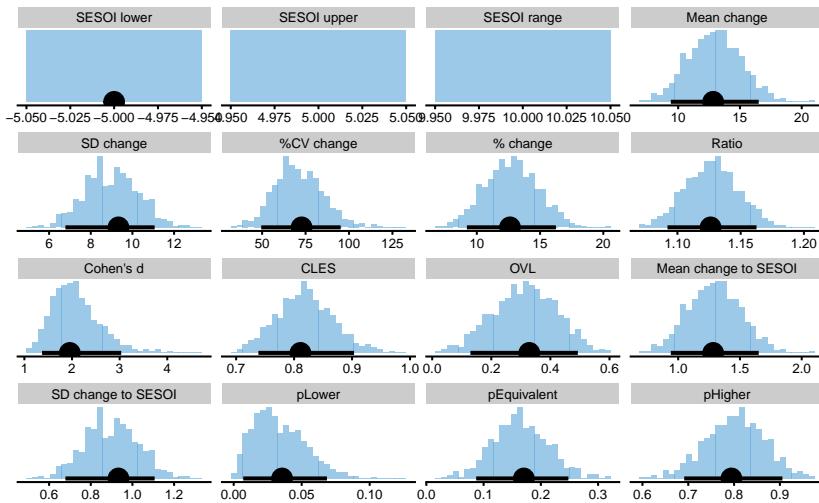
```

true_pre_post <- bmbstats::compare_dependent_groups(pre = bench_press_data$`Pre-test` ,
post = bench_press_data$`Post-test` (true) , SESOI_lower = -5,
SESOI_upper = 5, estimator_function = bmbstats::dependent_groups_estimators,
control = model_control(seed = 1667, boot_type = "perc",
boot_samples = 1000, confidence = 0.9))
#> [1] "All values of t are equal to 5 \n Cannot calculate confidence intervals"
#> [1] "All values of t are equal to 10 \n Cannot calculate confidence intervals"

```

```
true_pre_post
#> Bootstrap with 1000 resamples and 90% perc confidence intervals.
#>
#>           estimator      value      lower
#> SESOI lower -5.00000000 -5.00000000
#> SESOI upper  5.00000000      NA
#> SESOI range 10.00000000      NA
#> Mean change 12.82946286 9.402614340
#> SD change   9.32811444 6.781651565
#> %CV change 72.70853459 49.485375080
#> % change    12.62276799 9.213352403
#> Ratio       1.12622768 1.092133524
#> Cohen's d   1.95364779 1.371167791
#> CLES        0.81092721 0.738420828
#> OVL         0.32865634 0.129433313
#> Mean change to SESOI 1.28294629 0.940261434
#> SD change to SESOI 0.93281144 0.678165157
#> pLower     0.03558066 0.007004211
#> pEquivalent 0.17027690 0.087087118
#> pHIGHER    0.79414243 0.691105965
#>           upper
#> -5.00000000
#>      NA
#>      NA
#> 16.51981871
#> 11.06663336
#> 95.21126362
#> 16.25417940
#> 1.16254179
#> 3.03268427
#> 0.90342409
#> 0.49297559
#> 1.65198187
#> 1.10666334
#> 0.06858572
#> 0.24817063
#> 0.90546802
```

```
plot(true_pre_post)
```



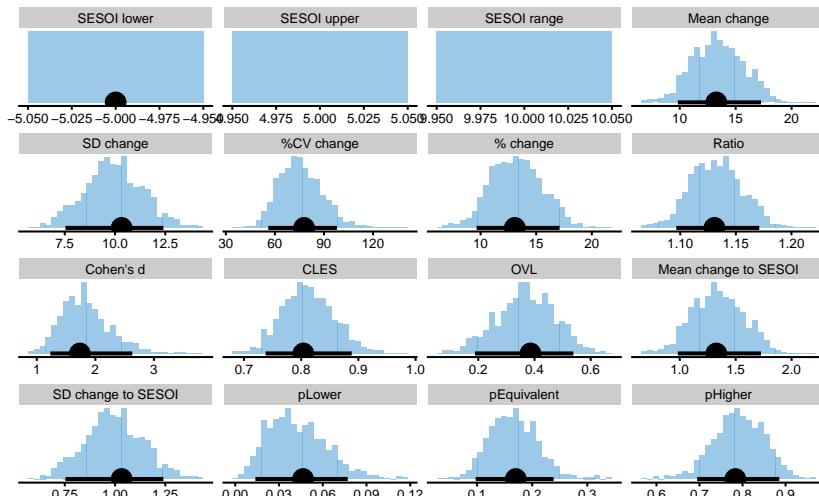
And now with the observed Pre- and Post- scores:

```
obs_pre_post <- bmbstats:::compare_dependent_groups(pre = bench_press_data$`Pre-test (observed)`, post = bench_press_data$`Post-test (observed)`, SESOI_lower = -5, SESOI_upper = 5, estimator_function = bmbstats:::dependent_groups_estimators, control = model_control(seed = 1667, boot_type = "perc", boot_samples = 1000, confidence = 0.9))
#> [1] "All values of t are equal to 5 \n Cannot calculate confidence intervals"
#> [1] "All values of t are equal to 10 \n Cannot calculate confidence intervals"

obs_pre_post
#> Bootstrap with 1000 resamples and 90% perc confidence intervals.
#>
#>           estimator      value      lower
#> SESOI lower -5.000000000 -5.000000000
#> SESOI upper  5.000000000   NA
#> SESOI range  10.000000000   NA
#> Mean change  13.26634977  9.82939542
#> SD change    10.32696439  7.53316566
#> %CV change   77.84329951  55.95245477
#> % change     13.07536952  9.64149847
#> Ratio        1.13075370  1.09641498
#> Cohen's d    1.73487556  1.23093674
#> CLES         0.80381999  0.73779303
#> OVL          0.38570219  0.18838477
#> Mean change to SESOI 1.32663498  0.98293954
#> SD change to SESOI  1.03269644  0.75331657
#> pLower       0.04648926  0.01380798
```

```
#>      pEquivalent 0.17017990 0.09737292
#>      pHiger 0.78333084 0.69453555
#>      upper
#> -5.00000000
#>      NA
#>      NA
#> 17.26567091
#> 12.40584025
#> 97.95459056
#> 17.10620410
#> 1.17106204
#> 2.63074534
#> 0.88834207
#> 0.53824551
#> 1.72656709
#> 1.24058402
#> 0.07714882
#> 0.23894195
#> 0.88551971
```

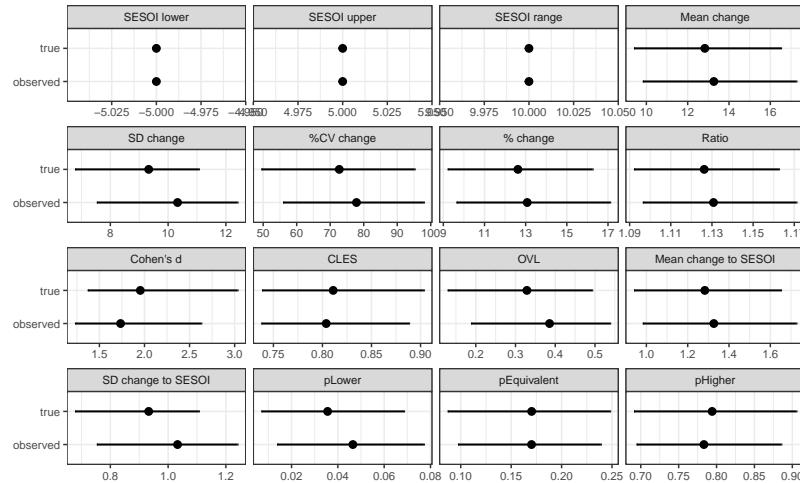
```
plot(obs_pre_post)
```



Let's plot the estimated CIs for all the estimators:

```
# Plot the estimators
compare_analyses <- rbind(data.frame(group = "true", true_pre_post$estimators),
  data.frame(group = "observed", obs_pre_post$estimators))
```

```
ggpplot(compare_analyses, aes(y = group, x = value)) + theme_bw(8) +
  geom_errorbarh(aes(xmax = upper, xmin = lower), color = "black",
                 height = 0) + geom_point() + xlab("") + ylab("") +
  facet_wrap(~estimator, scales = "free_x")
```



As can be seen on the figure, some estimators (those depending on the SD) are more affected by the measurement error. Since we do not know *true* scores, we can perform SIMEX analysis on the observed scores, or adjust SD of the change using the change measurement error. Let's do that by writing our own estimator equation that uses adjustment for the change SD:

```
adjusted_estimators <- function(pre, post, SESOI_lower = 0,
                                SESOI_upper = 0, na.rm = FALSE) {
  SESOI_range <- SESOI_upper - SESOI_lower
  change <- post - pre

  mean_change <- mean(change, na.rm = na.rm)
  sd_change <- stats::sd(change, na.rm = na.rm)

  # Now we adjust the sd_change with the known measurement
  # error
  change_measurement_error <- measurement_error * sqrt(2)

  sd_change <- sqrt(sd_change^2 - change_measurement_error^2)

  cv_change <- 100 * sd_change / mean_change
  perc_change <- mean(change / pre, na.rm = na.rm) * 100
```

```

ratio <- mean(post/pre, na.rm = na.rm)
cohen <- cohens_d(pre, post, paired = TRUE, na.rm = na.rm)
cles <- CLES(pre, post, na.rm = na.rm)
ovl <- 2 * stats::pnorm(-abs(cohen)/2)

change_to_SESOI <- mean_change/SESOI_range
sd_change_to_SESOI <- sd_change/SESOI_range

# Calculate proportion of scores
df <- length(change) - 1
higher <- 1 - stats::pt((SESOI_upper - mean_change)/sd_change,
                        df = df)
lower <- stats::pt((SESOI_lower - mean_change)/sd_change,
                     df = df)
equivalent <- 1 - (higher + lower)

c(`SESOI lower` = SESOI_lower, `SESOI upper` = SESOI_upper,
  `SESOI range` = SESOI_range, `Mean change` = mean_change,
  `SD change` = sd_change, `%CV change` = cv_change,
  `% change` = perc_change, Ratio = ratio, `Cohen's d` = cohen,
  CLES = cles, OVL = ovl, `Mean change to SESOI` = change_to_SESOI,
  `SD change to SESOI` = sd_change_to_SESOI, pLower = lower,
  pEquivalent = equivalent, pHIGHER = higher)
}

# -----
adj_pre_post <- bmbstats::compare_dependent_groups(pre = bench_press_data$`Pre-test (observed)`,
                                                    post = bench_press_data$`Post-test (observed)`,
                                                    SESOI_lower = -5,
                                                    SESOI_upper = 5, estimator_function = adjusted_estimators,
                                                    control = model_control(seed = 1667, boot_type = "perc",
                                                               boot_samples = 1000, confidence = 0.9))
#> [1] "All values of t are equal to 5 \n Cannot calculate confidence intervals"
#> [1] "All values of t are equal to 10 \n Cannot calculate confidence intervals"

adj_pre_post
#> Bootstrap with 1000 resamples and 90% perc confidence intervals.
#>
#>           estimator      value      lower
#> SESOI lower -5.000000000 -5.000000000
#> SESOI upper  5.000000000   NA
#> SESOI range 10.000000000   NA
#> Mean change 13.26634977  9.829395423
#> SD change   9.70289614  6.651960703
#> %CV change 73.13915511 50.327816706

```

```

#>           % change 13.07536952  9.641498467
#>           Ratio   1.13075370  1.096414985
#>           Cohen's d 1.73487556  1.230936741
#>           CLES    0.80381999  0.737793033
#>           OVL     0.38570219  0.188384768
#> Mean change to SESOI 1.32663498  0.982939542
#> SD change to SESOI  0.97028961  0.665196070
#>           pLower   0.03758233  0.007977625
#>           pEquivalent 0.16484497  0.085648118
#>           pHigher   0.79757270  0.708039046
#>           upper
#> -5.000000000
#>           NA
#>           NA
#> 17.26567091
#> 11.89137806
#> 93.01853927
#> 17.10620410
#> 1.17106204
#> 2.63074534
#> 0.88834207
#> 0.53824551
#> 1.72656709
#> 1.18913781
#> 0.06801839
#> 0.23506532
#> 0.90137633

```

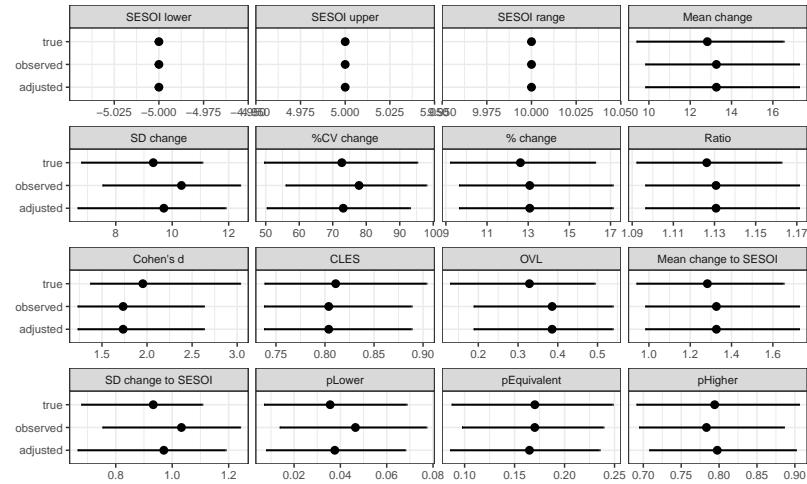
Now we can add these estimated CI to the graph and compare it with estimates using true and observed scores:

```

# Plot the estimators
compare_analyses <- rbind(data.frame(group = "true", true_pre_post$estimators),
                           data.frame(group = "observed", obs_pre_post$estimators),
                           data.frame(group = "adjusted", adj_pre_post$estimators))

ggplot(compare_analyses, aes(y = group, x = value)) + theme_bw(8) +
  geom_errorbarh(aes(xmax = upper, xmin = lower), color = "black",
                 height = 0) + geom_point() + xlab("") + ylab("") +
  facet_wrap(~estimator, scales = "free_x")

```



To be fair, some estimators like Cohen's d and those depending on it and Pre-test SD were not adjusted (which we can do that too as well), but SD change and other estimators dependent on that it we adjusted and much closer to the estimates using the true scores.

As explained in the [What to do when we know the error?](#) section, SIMEX procedure can be implemented as well.

This simple example demonstrates the effect of the measurement error on the estimators and a simple adjustment that could be done to come closer to estimators using true scores for the dependent group analysis. This is very similar to the RCT analysis, where SD of the Control group change scores will be used instead of known measurement error.

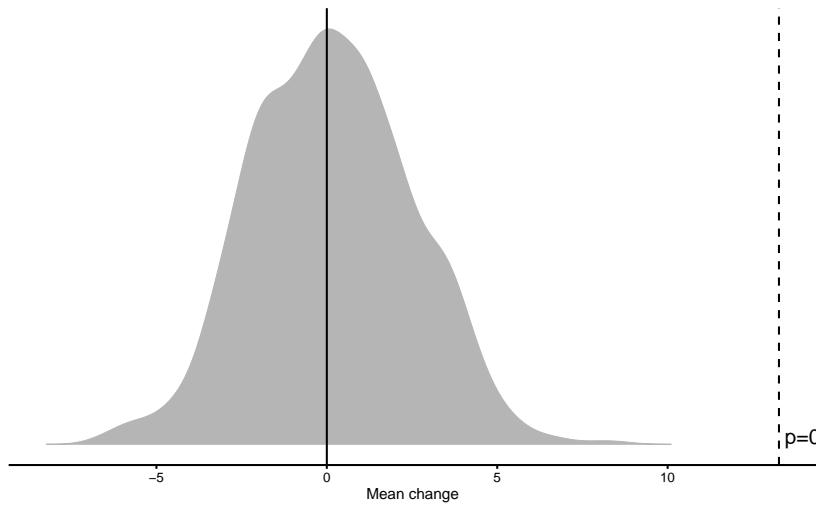
13.5.3 Statistical tests

For the sake of example, let's perform NHST using the `mean_change` estimator estimated using the observed scores:

```
pre_vs_post_NHST <- bmbstats::bootstrap_NHST(obs_pre_post,
  estimator = "Mean change", null_hypothesis = 0, test = "two.sided")

pre_vs_post_NHST
#> Null-hypothesis significance test for the `Mean change` estimator
#> Bootstrap result: Mean change=13.266, 90% CI [9.829, 17.266]
#> H0=0, test: two.sided
#> p=0.000999000999000999
```

```
plot(pre_vs_post_NHST)
```

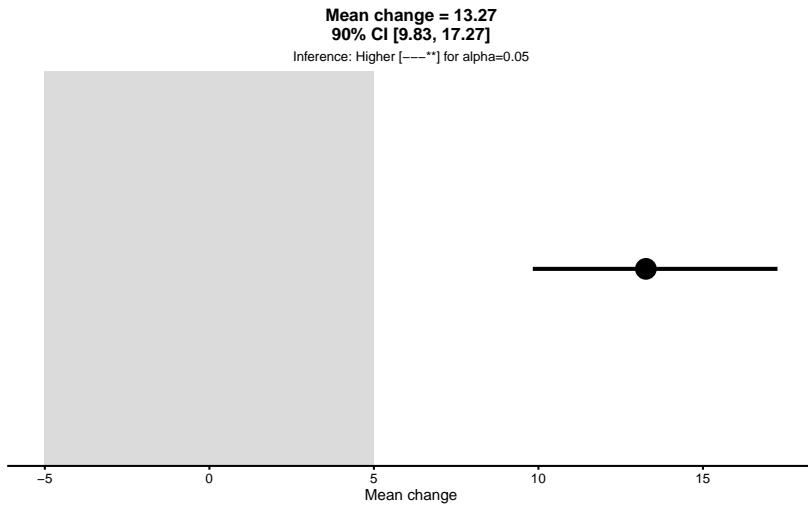


Using SESOI of $\pm 5\text{kg}$ for the `mean_change` estimator as well, let's do the METs:

```
pre_vs_post_MET <- bmbstats::bootstrap_MET(obs_pre_post,
  estimator = "Mean change", SESOI_lower = -5, SESOI_upper = 5,
  alpha = 0.05)

pre_vs_post_MET
#> Minimum effect tests for the `Mean change` estimator
#> Bootstrap result: Mean change=13.266, 90% CI [9.829, 17.266]
#> SESOI: [-5, 5], alpha=0.05
#>
#>           Test      p.value
#>   inferiority 1.0000000000
#> non-superiority 0.9990000000
#>     equivalence 0.9990000000
#> non-inferiority 0.000999001
#>    superiority 0.001000000
#>
#> Final inference: Higher

plot(pre_vs_post_MET)
```

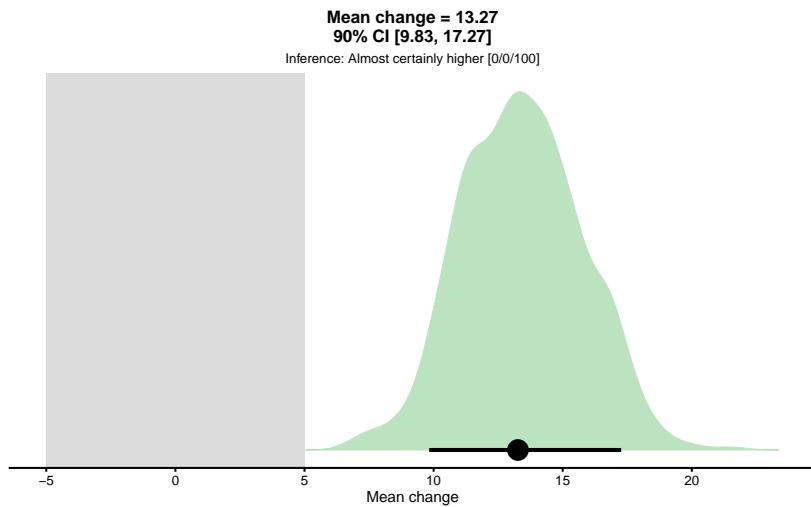


And finally MBI:

```
pre_vs_post_MBI <- bmbstats::bootstrap_MBI(obs_pre_post,
  estimator = "Mean change", SESOI_lower = -5, SESOI_upper = 5)

pre_vs_post_MBI
#> Magnitude-based inference for the `Mean change` estimator
#> Bootstrap result: Mean change=13.266, 90% CI [9.829, 17.266]
#> SESOI: [-5, 5]
#>
#>       Test prob
#>       lower     0
#>   equivalent     0
#>       higher    1
#>
#> Final inference: Almost certainly higher
```

```
plot(pre_vs_post_MBI)
```



13.6 Describing relationship between two groups

In [Describing relationship between two variables](#) section we have used a relationship between YoYoIR1 and MAS. This is how that data is generated, but without rounding (i.e. YoYoIR1 should be rounded to 40m and MAS to 0.5km/h):

```
set.seed(1667)

n_subjects <- 30

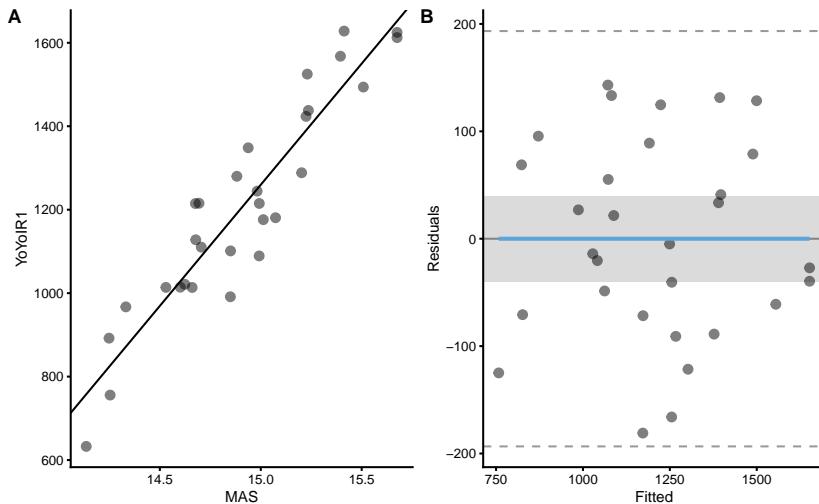
yoyo_mas_data <- tibble(Athlete = paste("Athlete", str_pad(string = seq(1,
  n_subjects), width = 2, pad = "0")), YoYoIR1 = rnorm(n = n_subjects,
  mean = 1224, sd = 255), MAS = 3.6 * (0.456 * YoYoIR1/1000 +
  3.617) + rnorm(n = length(YoYoIR1), 0, 0.2))

yoyo_mas_data
#> # A tibble: 30 x 3
#>   Athlete   YoYoIR1    MAS
#>   <chr>     <dbl> <dbl>
#> 1 Athlete 01    1628.  15.4
#> 2 Athlete 02    1089.  15.0
#> 3 Athlete 03    1438.  15.2
#> 4 Athlete 04    1215.  15.0
#> 5 Athlete 05     967.  14.3
#> 6 Athlete 06    1101.  14.9
```

```
#> 7 Athlete 07 1014. 14.5
#> 8 Athlete 08 1424. 15.2
#> 9 Athlete 09 633. 14.1
#> 10 Athlete 10 1348. 14.9
#> # ... with 20 more rows
```

Let's create a scatter plot with linear regression model using `bmbstats::plot_pair_lm`, with YoYoIR1 being outcome variable and MAS being predictor, with SESOI being ± 40 m

```
bmbstats::plot_pair_lm(predictor = yoyo_mas_data$MAS, outcome = yoyo_mas_data$YoYoIR1,
  predictor_label = "MAS", outcome_label = "YoYoIR1", SESOI_lower = -40,
  SESOI_upper = 40)
```



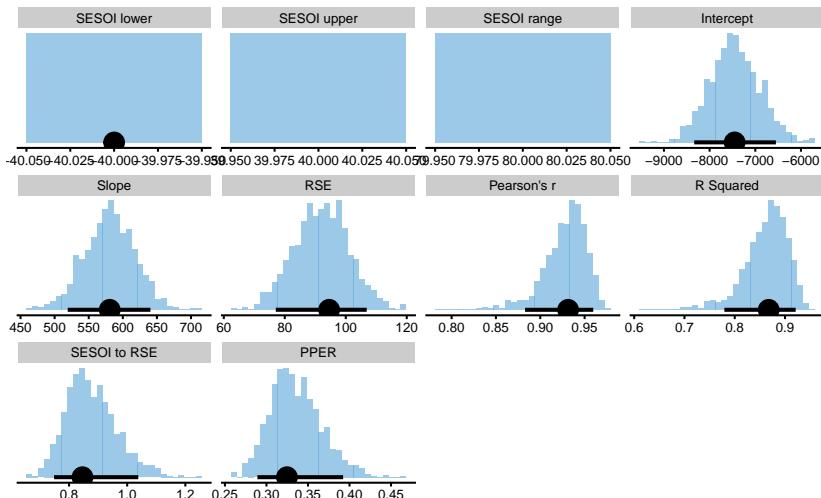
To get bootstrapped CIs of the estimators, use `bmbstats::describe_relationship` and `bmbstats::relationship_lm_estimators` functions:

```
boot_relationship <- bmbstats::describe_relationship(predictor = yoyo_mas_data$MAS,
  outcome = yoyo_mas_data$YoYoIR1, SESOI_lower = -40, SESOI_upper = 40,
  estimator_function = bmbstats::relationship_lm_estimators,
  control = model_control(seed = 1667, boot_type = "perc",
  boot_samples = 1000, confidence = 0.9))
#> [1] "All values of t are equal to 40 \n Cannot calculate confidence intervals"
#> [1] "All values of t are equal to 80 \n Cannot calculate confidence intervals"

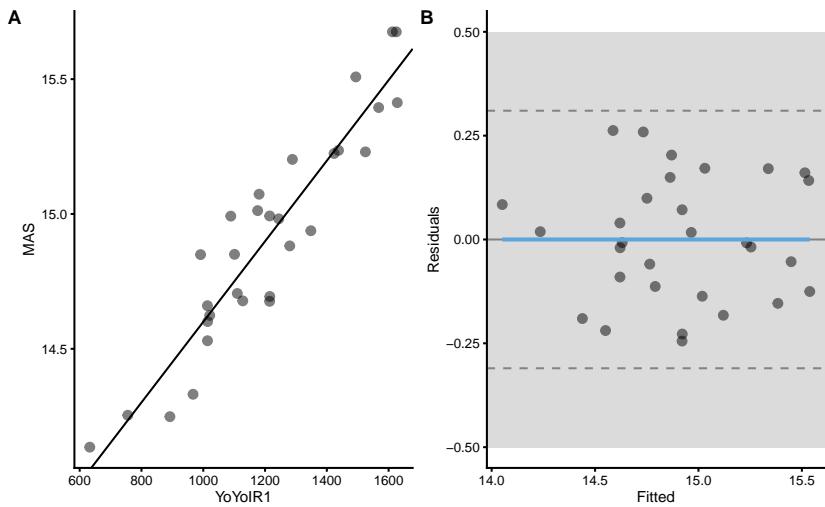
boot_relationship
#> Bootstrap with 1000 resamples and 90% perc confidence intervals.
```

```
#>
#>   estimator      value      lower      upper
#> SESOI lower -40.0000000 -40.0000000 -40.0000000
#> SESOI upper 40.0000000          NA          NA
#> SESOI range 80.0000000          NA          NA
#> Intercept -7452.2295288 -8337.2111534 -6549.1633775
#> Slope      580.8003907  519.2080676  640.5107552
#> RSE        94.5402638  77.0626151 106.8726878
#> Pearson's r 0.9314150  0.8828261 0.9597758
#> R Squared   0.8675339  0.7793820 0.9211695
#> SESOI to RSE 0.8462003  0.7485542 1.0381170
#> PPER       0.3246566  0.2890790 0.3923412
```

```
plot(boot_relationship)
```



Magnitude-based estimators **SESOI to RSE** and **PPER** are useful in judging practical significance of this model, which is this case very bad. For example, if we now use MAS as outcome and YoYoIR1 as predictor with SESOI equal to $\pm 0.5\text{kmh}$, R Squared and Pearson's r will stay the same, but **SESOI to RSE** and **PPER** will demonstrate model that now had much better practical significance:



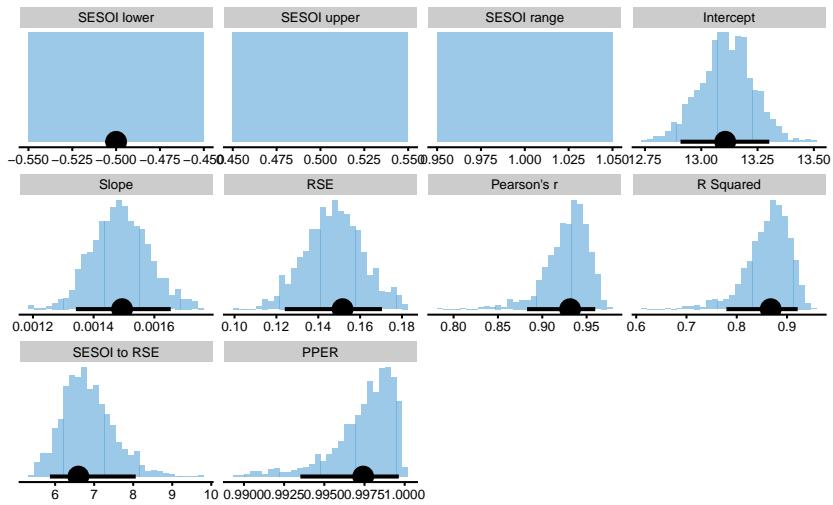
```

boot_relationship <- bmbstats::describe_relationship(outcome = yoyo_mas_data$MAS,
  predictor = yoyo_mas_data$YoYoIR1, SESOI_lower = -0.5,
  SESOI_upper = 0.5, estimator_function = bmbstats::relationship_lm_estimators,
  control = model_control(seed = 1667, boot_type = "perc",
  boot_samples = 1000, confidence = 0.9))
#> [1] "All values of t are equal to 0.5 \n Cannot calculate confidence intervals"
#> [1] "All values of t are equal to 1 \n Cannot calculate confidence intervals"

boot_relationship
#> Bootstrap with 1000 resamples and 90% perc confidence intervals.
#>
#>   estimator      value      lower      upper
#>   SESOI lower -0.5000000000 -0.5000000000 -0.5000000000
#>   SESOI upper  0.5000000000          NA          NA
#>   SESOI range   1.0000000000          NA          NA
#>   Intercept  13.106232758 12.90779375 13.302068612
#>   Slope     0.001493687  0.00134178  0.001654143
#>   RSE       0.151611849  0.12399493  0.170447293
#>   Pearson's r 0.931415014  0.88282614  0.959775759
#>   R Squared  0.867533928  0.77938199  0.921169508
#>   SESOI to RSE 6.595790544  5.86691658  8.064845693
#>   PPER      0.997419105  0.99351216  0.999633678

plot(boot_relationship)

```



In the next chapter we will continue with the prediction tasks.

Chapter 14

Predictive tasks using **bmbstats**

Simple implementation of the prediction tasks in **bmbstats** is done with **bmbstats::cv_model** function, which is short of “cross-validate model”. This function is more of a teaching tool, than something more thorough like the **caret** (Kuhn 2020) or **mlr** and **mlr3** packages (Bischl et al. 2016; Lang et al. 2019). But **bmbstats::cv_model** is still powerful and versatile for the sports science prediction tasks (e.g. **bmbstats::rct_predict** is a function that we will introduce later and it was built using **bmbstats::cv_model** results). **bmbstats::cv_model** is built using the outstanding **hardhat** package (Vaughan and Kuhn 2020).

bmbstats::cv_model has three components: (1) fitting or modeling function (set using the **model_func** parameter with the **bmbstats::lm_model**(default) and **bmbstats::baseline_model** being implemented using **stats::lm** linear regression function), (2) prediction function; which is used for predicting on the new and training data (set using the **predict_func** parameter with **bmbstats::generic_predict** being the default that calls the default predict method, or **stats::predict.lm** for the **lm** classes), and (3) performance function; which is used to return performance estimators (set using the **perf_func** parameter with **bmbstats::performance_metrics** being the default). This will be much clearer once we start implementing the **bmbstats::cv_model**.

To demonstrate **bmbstats::cv_model** function, let's generate simple data for prediction:

```
require(tidyverse)
require(bmbstats)
require(cowplot)
```

```

set.seed(1667)

# Model (DGP)
random_error <- 2

sinus_data <- tibble(x = seq(0.8, 2.5, 0.05), observed_y = 30 +
  15 * (x * sin(x)) + rnorm(n = length(x), mean = 0, sd = random_error),
  true_y = 30 + 15 * (x * sin(x)))

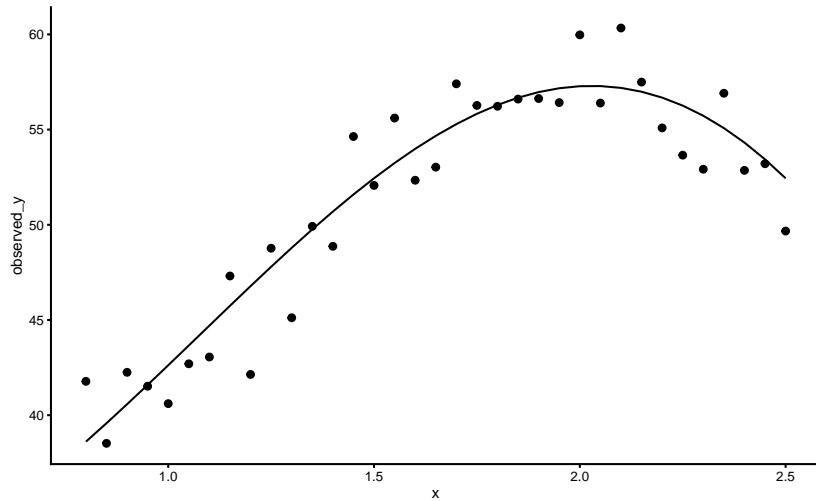
head(sinus_data)
#> # A tibble: 6 x 3
#>   x     observed_y   true_y
#>   <dbl>      <dbl>    <dbl>
#> 1 0.8        41.8    38.6
#> 2 0.85       38.5    39.6
#> 3 0.9        42.3    40.6
#> 4 0.95       41.5    41.6
#> 5 1          40.6    42.6
#> 6 1.05       42.7    43.7

```

```

ggplot(sinus_data, aes(x = x)) + theme_cowplot(8) + geom_point(aes(y = observed_y)) +
  geom_line(aes(y = true_y))

```



This data has *irreducible error* with SD equal to 2 (a.u.). Let's use simple linear regression to predict observed y , but evaluate model performance using 10 repeats of 5-folds cross-validations:

```

model1 <- bmbstats::cv_model(
  observed_y ~ x,
  sinus_data,

  # These are default options, but I will list them here
  model_func = lm_model,
  predict_func = generic_predict,
  perf_func = performance_metrics,

  # CV parameters
  control = model_control(
    cv_folds = 5,
    cv_repeats = 10,
    seed = 1667
  )
)

model1
#> Training data consists of 2 predictors and 35 observations. Cross-Validation of the model was
#>
#> Model performance:
#>
#>           metric      training training.pooled
#>           MBE -1.786530e-14   3.608730e-15
#>           MAE  3.307366e+00   3.281275e+00
#>           RMSE 3.830961e+00   3.811531e+00
#>           PPER  2.500597e-01   2.532192e-01
#> SESOI to RMSE 6.520075e-01   6.461322e-01
#>           R-squared 6.125604e-01   6.164804e-01
#>           MinErr -5.833192e+00   -6.481721e+00
#>           MaxErr  9.531759e+00   1.069301e+01
#>           MaxAbsErr 9.531759e+00   1.069301e+01
#> testing.pooled      mean        SD       min
#> 0.01201151 0.03878805 1.39091047 -3.56918463
#> 3.42055845 3.40542550 0.71086848  1.90802477
#> 4.00171168 3.91431323 0.78174879  2.19948265
#> 0.24118080 0.22872198 0.05453147  0.15058656
#> 0.61542490 0.67035639 0.16588651  0.47111531
#> 0.57725715 0.59398198 0.20141946 -0.07623944
#> -6.66660654 -4.83585586 1.34488218 -6.66660654
#> 10.89240388 5.74134976 2.57512537  1.06931471
#> 10.89240388 6.54646863 1.91850684  3.68446911
#>           max
#> 2.8362928
#> 4.6087168

```

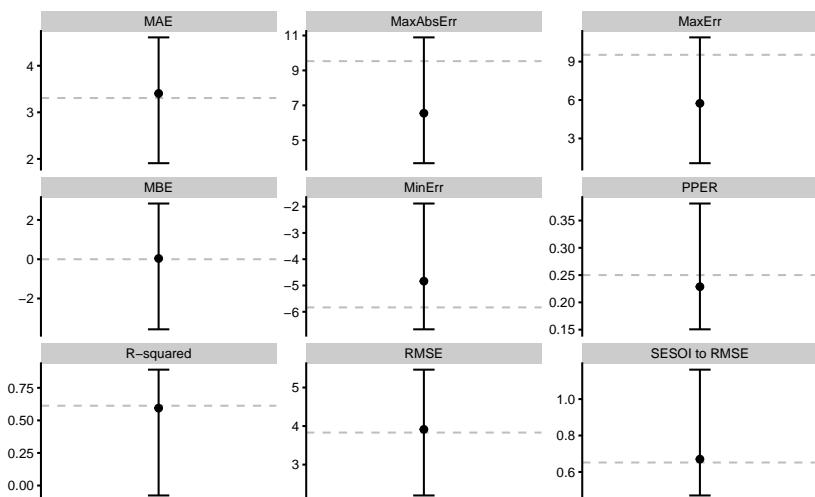
```
#> 5.4637498
#> 0.3811788
#> 1.1614533
#> 0.8893906
#> -1.8826434
#> 10.8924039
#> 10.8924039
```

SESOI in the above example is calculated using $0.2 \times \text{SD}$ of the outcome variable (i.e. `observed_y`) which represents Cohen's trivial effect. SESOI constants of estimation function (that uses training data set to estimate SESOI) can be set up using the `SESOI_lower` and `SESOI_upper` parameters (to which the default are `bmbstats::SESOI_lower_func` and `bmbstats::SESOI_upper_func` respectively).

The above output represents performance summary using the estimators returned by the `bmbstats::performance_metrics`. `bmbstats` has numerous *cost* functions implemented that could be called using the `bmbstats::cost_` prefix.

The above output can be plotted using the `plot` command and `type = "estimators"` parameter:

```
plot(model1, "estimators")
```



Error bars represent range of estimator values across cross-validation folds, while the dashed line indicate training performance. These estimates can be accessed in the returned object, i.e. `model1$performance` for training, and `model1$cross_validation$performance` for cross-validation.

14.1 How to implement different performance metrics?

To implement different performance metrics you need to write your own function that return named vector using the following template:

```
# My performance metrics
my_perf_metrics <- function(observed,
                             predicted,
                             SESOI_lower = 0,
                             SESOI_upper = 0,
                             na.rm = FALSE) {
  c(
    RMSE = cost_RMSE(
      observed = observed,
      predicted = predicted,
      SESOI_lower = SESOI_lower,
      SESOI_upper = SESOI_upper,
      na.rm = na.rm
    ),
    PPER = cost_PPER(
      observed = observed,
      predicted = predicted,
      SESOI_lower = SESOI_lower,
      SESOI_upper = SESOI_upper,
      na.rm = na.rm
    ),
    `R-squared` = cost_R_squared(
      observed = observed,
      predicted = predicted,
      SESOI_lower = SESOI_lower,
      SESOI_upper = SESOI_upper,
      na.rm = na.rm
    )
  )
}
```

```
# Re-run the cv_model with my perf metrics
model2 <- bmbstats::cv_model(
  observed_y ~ x,
  sinus_data,
```

```

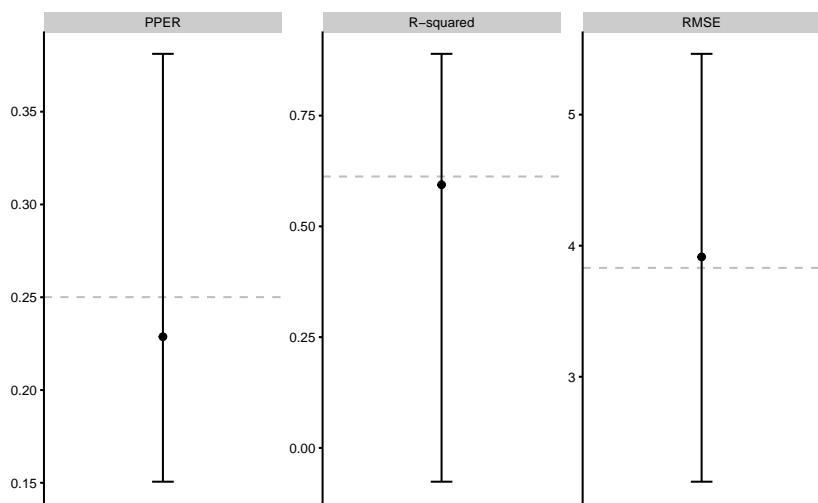
# Use our performance metrics
perf_func = my_perf_metrics,

# CV parameters
control = model_control(
  cv_folds = 5,
  cv_repeats = 10,
  seed = 1667
)
)

model2
#> Training data consists of 2 predictors and 35 observations. Cross-Validation of the
#>
#> Model performance:
#>
#>      metric training training.pooled testing.pooled
#>      RMSE 3.8309607      3.8115312      4.0017117
#>      PPER 0.2500597      0.2532192      0.2411808
#>      R-squared 0.6125604      0.6164804      0.5772572
#>      mean          SD          min          max
#> 3.914313 0.78174879 2.19948265 5.4637498
#> 0.228722 0.05453147 0.15058656 0.3811788
#> 0.593982 0.20141946 -0.07623944 0.8893906

```

plot(model2, "estimators")



14.2 How to use different prediction model?

To use different prediction model instead of `stats::lm`, you will need to modify the model function using the following template. Let's use Regression Tree using the `rpart` package (Therneau and Atkinson 2019):

```
require(rpart)

# My prediction model
my_model_func <- function(predictors,
                           outcome,
                           SESOI_lower = 0,
                           SESOI_upper = 0,
                           na.rm = FALSE,
                           ...) {
  data <- cbind(.outcome = outcome, predictors)
  rpart(.outcome ~ ., data = data, ...)
}

# Call the cv_model
model3 <- bmbstats::cv_model(
  observed_y ~ x,
  sinus_data,

  # Use our model function
  model_func = my_model_func,

  # CV parameters
  control = model_control(
    cv_folds = 5,
    cv_repeats = 10,
    seed = 1667
  ),

  # Do not create intercept column
  intercept = TRUE
)

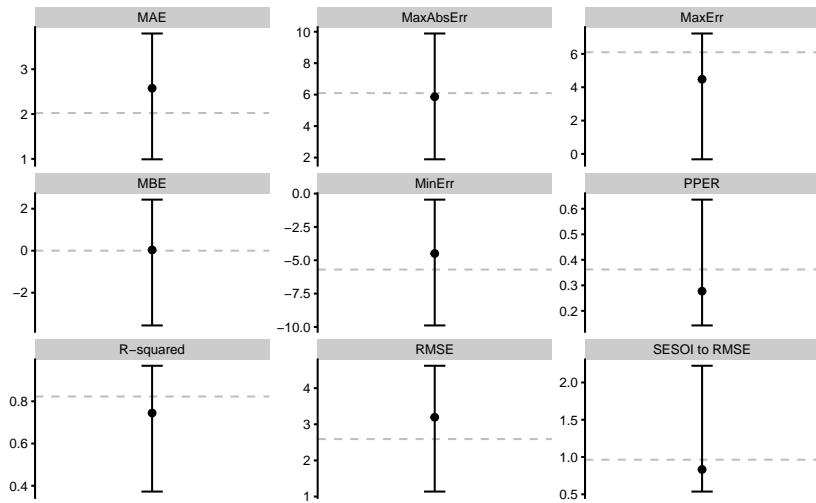
model3
#> Training data consists of 2 predictors and 35 observations. Cross-Validation of the model was
#>
#> Model performance:
#>
#>      metric      training training.pooled
#>          MBE  2.030618e-16 -3.045865e-16
```

```

#>      MAE  2.023389e+00  2.268546e+00
#>      RMSE 2.591359e+00  2.765355e+00
#>      PPER 3.621870e-01  3.437038e-01
#> SESOI to RMSE 9.639014e-01  8.905739e-01
#>      R-squared 8.227265e-01  7.981214e-01
#>      MinErr -5.699018e+00 -6.168662e+00
#>      MaxErr  6.097584e+00  6.427283e+00
#>      MaxAbsErr 6.097584e+00  6.427283e+00
#> testing.pooled      mean      SD      min
#> 0.03760974 0.03641386 1.38472089 -3.5600378
#> 2.60389737 2.57549008 0.60425150  0.9905252
#> 3.29362285 3.19456487 0.67699316  1.1382304
#> 0.29087228 0.27713101 0.07822901  0.1429604
#> 0.74773376 0.83310652 0.26906599  0.5367524
#> 0.71366153 0.74453263 0.13595473  0.3724261
#> -9.88715278 -4.49333669 2.25968230 -9.8871528
#> 7.22167137 4.47699323 1.95348134 -0.3225821
#> 9.88715278 5.86162382 1.31129516  1.8902345
#>      max
#> 2.4319242
#> 3.7964691
#> 4.6196434
#> 0.6357025
#> 2.2239463
#> 0.9682861
#> -0.4531138
#> 7.2216714
#> 9.8871528

```

```
plot(model3, "estimators")
```



14.3 Example of using tuning parameter

Let's use the same example from the [Overfitting](#) section of the [Prediction](#) chapter - polynomial fit. The objective is to select the polynomial degree that gives the best cross-validation performance.

```

poly_tuning <- seq(1, 12)

my_model_func <- function(predictors,
                           outcome,
                           SESOI_lower = 0,
                           SESOI_upper = 0,
                           na.rm = FALSE,
                           poly_n = 1) {
  data <- cbind(.outcome = outcome, predictors)
  lm(.outcome ~ poly(x, poly_n), data = data)
}

# Model performance across different tuning parameters
poly_perf <- map_df(poly_tuning, function(poly_n) {
  model <- bmbstats::cv_model(
    observed_y ~ x,
    sinus_data,
    # CV parameters
    control = model_control(
      cv_folds = 5,

```

```

    cv_repeats = 10,
    seed = 1667
),
model_func = my_model_func,
poly_n = poly_n
)

data.frame(
  poly_n = poly_n,
  model$cross_validation$performance$summary$overall
)
})

head(poly_perf)
#>   poly_n      metric   training training.pooled
#> 1     1       MBE -1.339861e-14  1.081218e-15
#> 2     1       MAE  3.307366e+00  3.281275e+00
#> 3     1       RMSE 3.830961e+00  3.811531e+00
#> 4     1       PPER 2.500597e-01  2.532192e-01
#> 5     1 SESOI to RMSE 6.520075e-01  6.461322e-01
#> 6     1 R-squared 6.125604e-01  6.164804e-01
#>   testing.pooled      mean        SD        min
#> 1     0.01201151 0.03878805 1.39091047 -3.56918463
#> 2     3.42055845 3.40542550 0.71086848  1.90802477
#> 3     4.00171168 3.91431323 0.78174879  2.19948265
#> 4     0.24118080 0.22872198 0.05453147  0.15058656
#> 5     0.61542490 0.67035639 0.16588651  0.47111531
#> 6     0.57725715 0.59398198 0.20141946 -0.07623944
#>   max
#> 1 2.8362928
#> 2 4.6087168
#> 3 5.4637498
#> 4 0.3811788
#> 5 1.1614533
#> 6 0.8893906

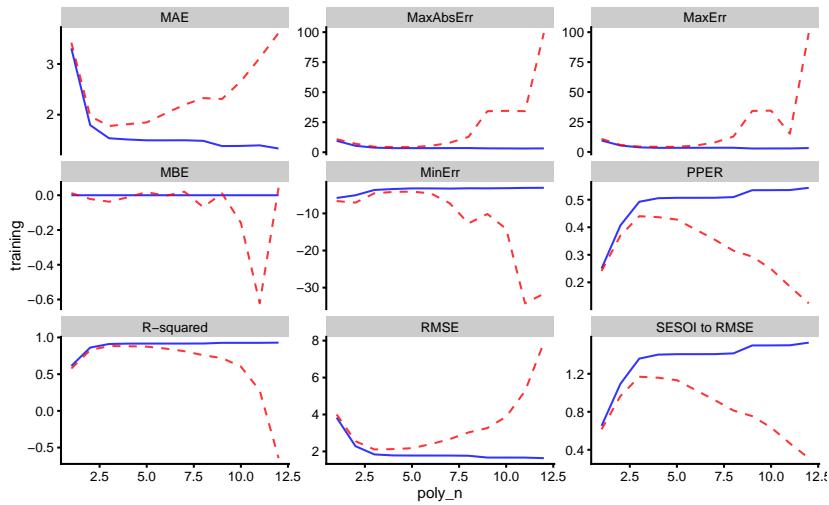
```

In the figure below the results of this analysis is depicted. Dashed red line represents cross-validated performance (using performance on the pooled testing data).

```

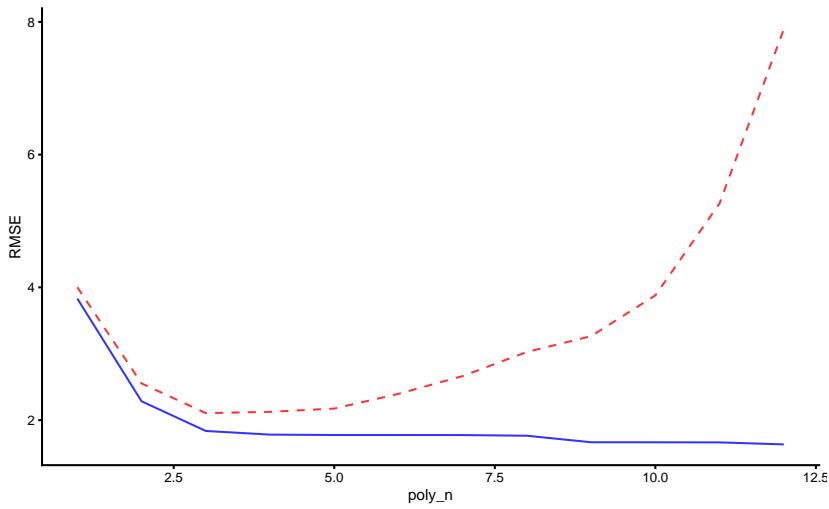
ggplot(poly_perf, aes(x = poly_n)) + theme_cowplot(8) + geom_line(aes(y = training),
  color = "blue", alpha = 0.8) + geom_line(aes(y = testing.pooled),
  color = "red", linetype = "dashed", alpha = 0.8) + facet_wrap(~metric,
  scales = "free_y")

```



As can be seen, the best predictive performance is with 3rd polynomial degrees. The figure below depicts RMSE estimator for higher resolution image.

```
ggplot(filter(poly_perf, metric == "RMSE"), aes(x = poly_n)) +
  theme_cowplot(8) + geom_line(aes(y = training), color = "blue",
  alpha = 0.8) + geom_line(aes(y = testing.pooled), color = "red",
  linetype = "dashed", alpha = 0.8) + ylab("RMSE")
```



14.4 Plotting

Various diagnostic graphs can be easily generated using a generic `plot` function. Let's fit a 3rd degree polynomial model first:

```
model4 <- bmbstats::cv_model(
  observed_y ~ poly(x, 3),
  sinus_data,

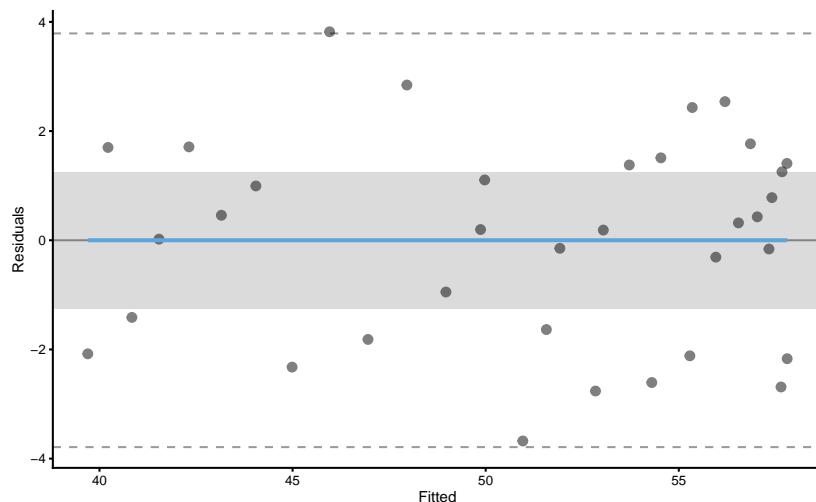
  # CV parameters
  control = model_control(
    cv_folds = 5,
    cv_repeats = 10,
    seed = 1667
  )
)

model4
#> Training data consists of 4 predictors and 35 observations. Cross-Validation of the
#>
#> Model performance:
#>
#>      metric      training training.pooled
#>      MBE -1.299275e-14   2.060564e-15
#>      MAE  1.534398e+00   1.509623e+00
#>      RMSE 1.837539e+00   1.807043e+00
#>      PPER  4.925435e-01   5.041349e-01
#> SESOI to RMSE 1.359326e+00   1.362863e+00
#> R-squared 9.108623e-01   9.137963e-01
#> MinErr -3.676547e+00   -4.335315e+00
#> MaxErr  3.819068e+00   4.380287e+00
#> MaxAbsErr 3.819068e+00   4.380287e+00
#> testing.pooled      mean          SD          min
#> -0.03724767 -0.02930855 0.83265732 -2.3250491
#> 1.771622691  1.77362879 0.33144744  1.0364816
#> 2.10648622  2.07925558 0.29737351  1.3940723
#> 0.44022003  0.40113906 0.05824287  0.2221031
#> 1.16912846  1.22806404 0.18072558  0.9198745
#> 0.88289646  0.89017328 0.04568934  0.7776291
#> -4.57284164 -2.99135803 0.81846093 -4.5728416
#> 4.43506709  2.54927044 1.17063338 -0.2033888
#> 4.57284164  3.45414085 0.57631575  2.3829256
#> max
#> 1.3681763
#> 2.3883585
```

```
#> 2.7092065  
#> 0.5513211  
#> 1.7145206  
#> 0.9626275  
#> 0.2782154  
#> 4.4350671  
#> 4.5728416
```

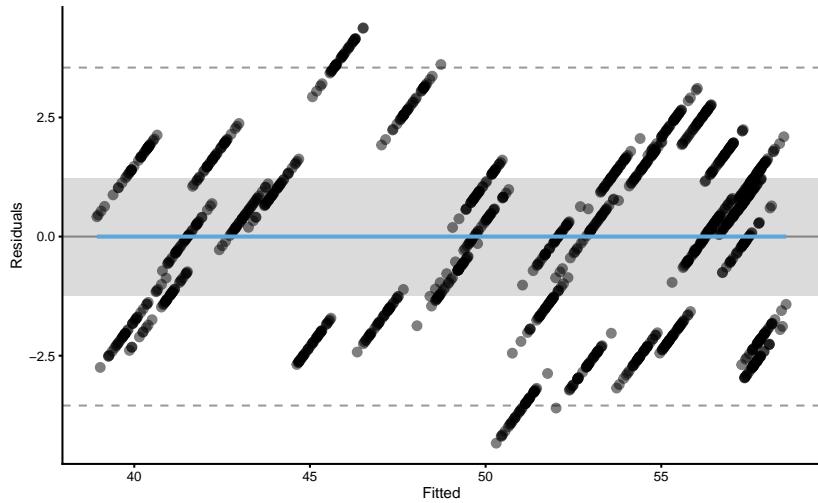
To plot the residuals, use `type="residuals"` in the `plot` function:

```
plot(model4, "residuals")
```

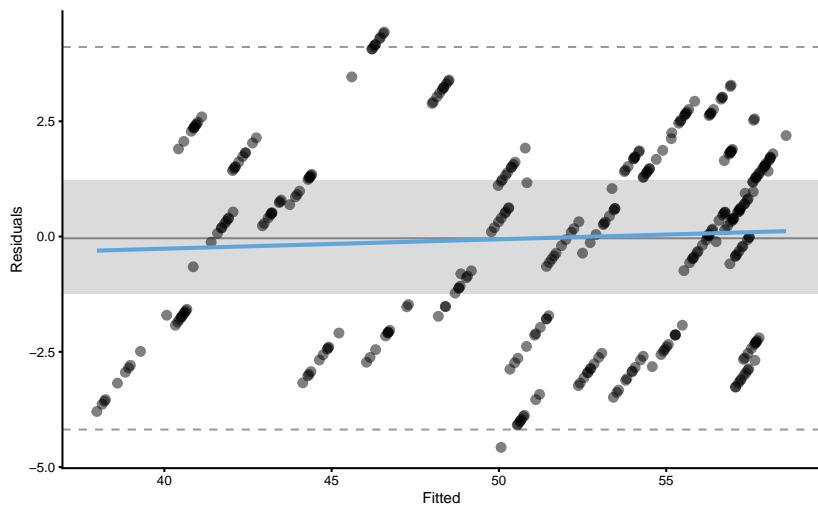


The following code will plot the training and testing residuals across cross-validation folds:

```
plot(model4, "training-residuals")
```

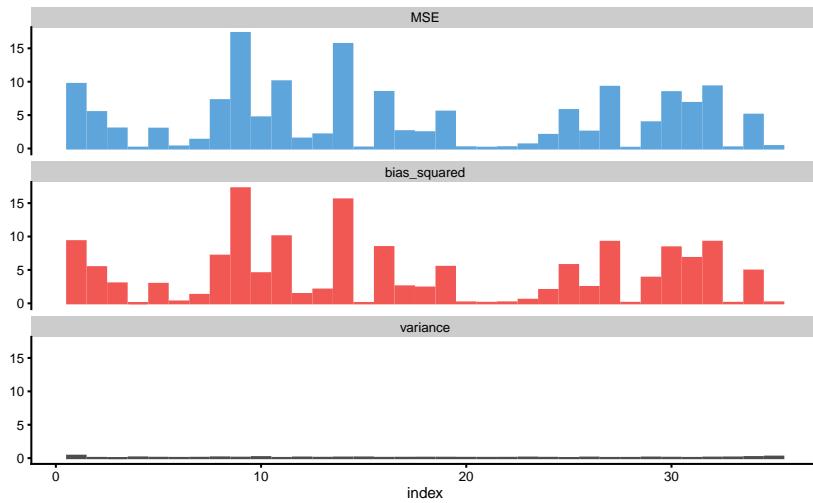


```
plot(model4, "testing-residuals")
```



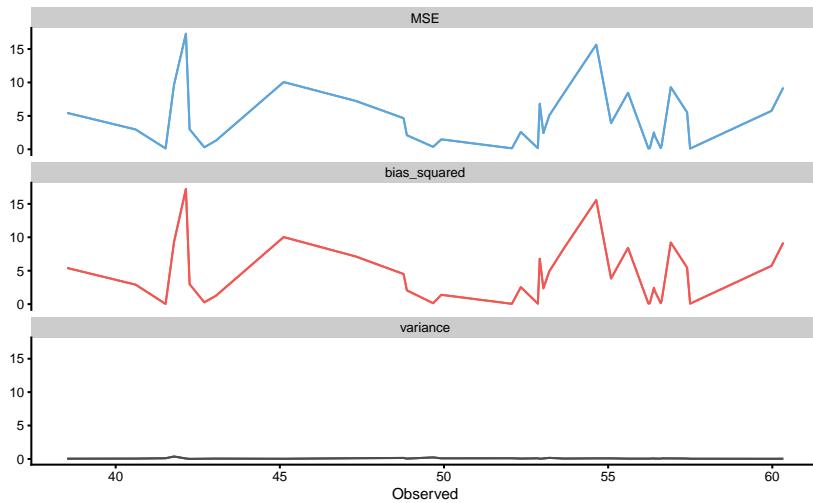
To plot bias-variance error decomposition across cross-validation folds use:

```
plot(model4, "bias-variance-index")
```



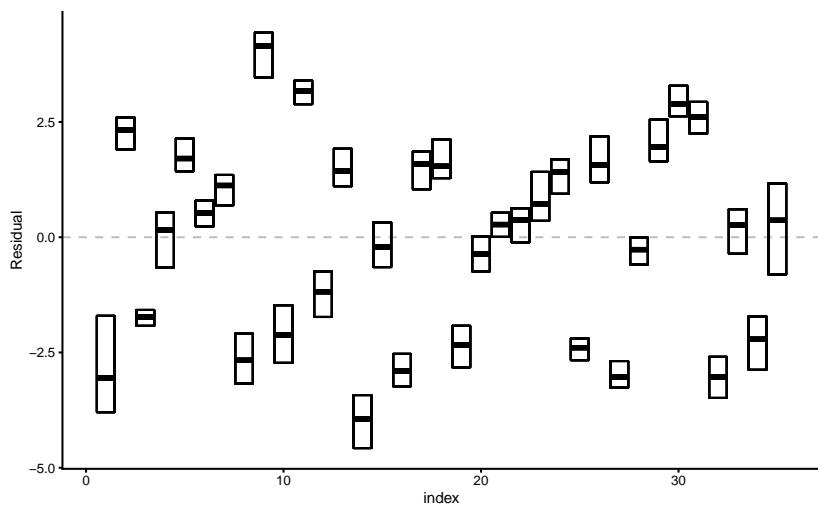
The above figure depicts bias-variance for each observation index. To plot against the outcome variable value use:

```
plot(model4, "bias-variance-observed")
```



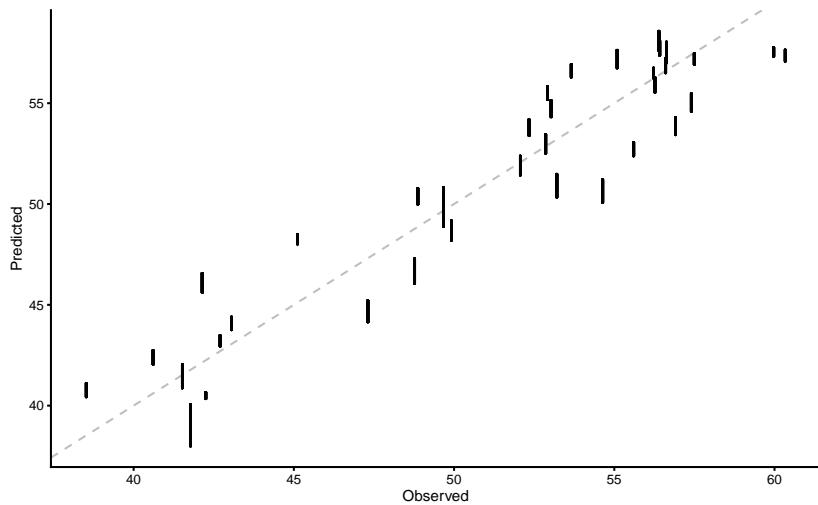
Prediction error (i.e. residuals) can also be plotted as boxes, demonstrating the mean (i.e. bias) and spread (i.e. variance). To plot prediction error again outcome index use:

```
plot(model4, "prediction-index")
```



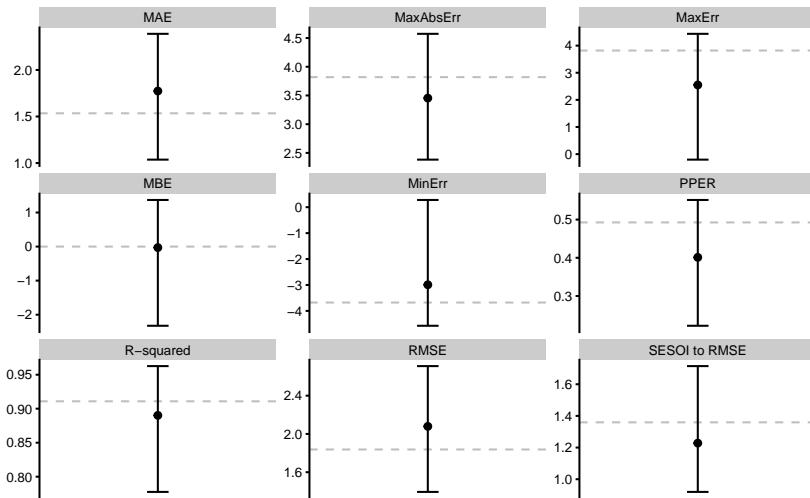
Similar to plotting bias-variance, prediction error distribution (actually, a spread or range) can be plotted against outcome variable value:

```
plot(model4, "prediction-observed")
```



And finally, to plot the performance estimator use:

```
plot(model4, "estimators")
```



Each of these plots can be customized using the `control=bmbstats::plot_control` argument and function.

14.5 Comparing models

Comparing models is beyond the scope of this book, but I will provide a short introduction using `bmbstats`. Let's use our `model3` that used `rpart` Regression Tree, and `model4` that used 3rd degree polynomial fit, to plot the estimators range across cross-validation folds.

```
plot_data <- rbind(data.frame(model = "rpart", model3$cross_validation$performance$summary$overall)
                     data.frame(model = "poly", model4$cross_validation$performance$summary$overall))

plot_data
#>   model      metric    training training.pooled
#> 1 rpart      MBE  2.030618e-16 -3.045865e-16
#> 2 rpart      MAE  2.023389e+00  2.268546e+00
#> 3 rpart      RMSE 2.591359e+00  2.765355e+00
#> 4 rpart      PPER 3.621870e-01  3.437038e-01
#> 5 rpart  SESOI to RMSE 9.639014e-01  8.905739e-01
#> 6 rpart      R-squared 8.227265e-01  7.981214e-01
#> 7 rpart      MinErr -5.699018e+00 -6.168662e+00
#> 8 rpart      MaxErr  6.097584e+00  6.427283e+00
#> 9 rpart      MaxAbsErr 6.097584e+00  6.427283e+00
```

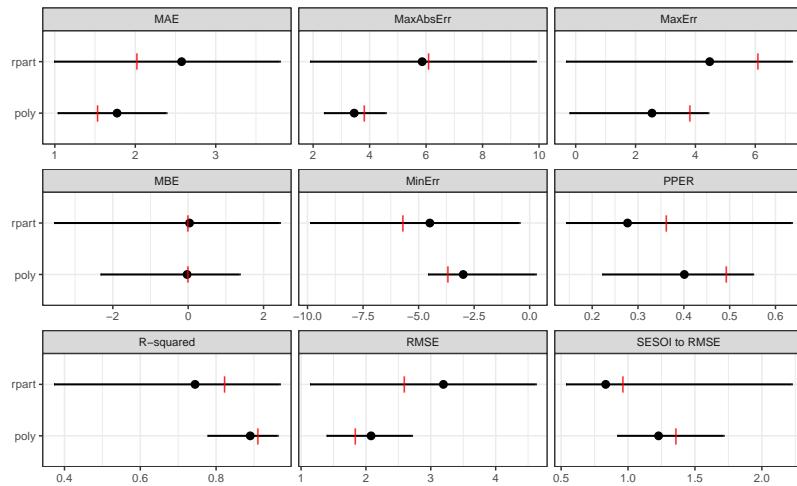
```

#> 10  poly          MBE -1.299275e-14  2.060564e-15
#> 11  poly          MAE  1.534398e+00  1.509623e+00
#> 12  poly          RMSE 1.837539e+00  1.807043e+00
#> 13  poly          PPER 4.925435e-01  5.041349e-01
#> 14  poly SESOI to RMSE 1.359326e+00  1.362863e+00
#> 15  poly          R-squared 9.108623e-01  9.137963e-01
#> 16  poly          MinErr -3.676547e+00 -4.335315e+00
#> 17  poly          MaxErr  3.819068e+00  4.380287e+00
#> 18  poly          MaxAbsErr 3.819068e+00  4.380287e+00
#>   testing.pooled      mean        SD       min
#> 1    0.03760974  0.03641386 1.38472089 -3.5600378
#> 2    2.60389737  2.57549008 0.60425150  0.9905252
#> 3    3.29362285  3.19456487 0.67699316  1.1382304
#> 4    0.29087228  0.27713101 0.07822901  0.1429604
#> 5    0.74773376  0.83310652 0.26906599  0.5367524
#> 6    0.71366153  0.74453263 0.13595473  0.3724261
#> 7    -9.88715278 -4.49333669 2.25968230 -9.8871528
#> 8    7.22167137  4.47699323 1.95348134 -0.3225821
#> 9    9.88715278  5.86162382 1.31129516  1.8902345
#> 10   -0.03724767 -0.02930855 0.83265732 -2.3250491
#> 11   1.77622691  1.77362879 0.33144744  1.0364816
#> 12   2.10648622  2.07925558 0.29737351  1.3940723
#> 13   0.44022003  0.40113906 0.05824287  0.2221031
#> 14   1.16912846  1.22806404 0.18072558  0.9198745
#> 15   0.88289646  0.89017328 0.04568934  0.7776291
#> 16   -4.57284164 -2.99135803 0.81846093 -4.5728416
#> 17   4.43506709  2.54927044 1.17063338 -0.2033888
#> 18   4.57284164  3.45414085 0.57631575  2.3829256
#>   max
#> 1    2.4319242
#> 2    3.7964691
#> 3    4.6196434
#> 4    0.6357025
#> 5    2.2239463
#> 6    0.9682861
#> 7    -0.4531138
#> 8    7.2216714
#> 9    9.8871528
#> 10   1.3681763
#> 11   2.3883585
#> 12   2.7092065
#> 13   0.5513211
#> 14   1.7145206
#> 15   0.9626275
#> 16   0.2782154

```

```
#> 17 4.4350671
#> 18 4.5728416
```

```
ggplot(plot_data, aes(y = model, x = mean)) + theme_bw(8) +
  geom_errorbarh(aes(xmax = max, xmin = min), color = "black",
                 height = 0) + geom_point() + geom_point(aes(x = training),
                 color = "red", shape = "|", size = 3) + xlab("") + ylab("") +
  facet_wrap(~metric, scales = "free_x")
```



As can be seen from the figure, polynomial fit represents a more predictive model. But we can also perform statistical significance test using the cross-validation performance, for let's say RMSE estimator. Since the CV folds are the same (which we achieved by using the same `seed` number), we can do dependent groups analysis. But before, let's plot the CV estimates of the RMSE:

```
rmse_rpart <- filter(model3$cross_validation$performance$folds$testing,
                      metric == "RMSE")
head(rmse_rpart)
#>           fold metric    value
#> RMSE...1 Fold1.Rep01  RMSE 3.535807
#> RMSE...2 Fold2.Rep01  RMSE 1.138230
#> RMSE...3 Fold3.Rep01  RMSE 3.175505
#> RMSE...4 Fold4.Rep01  RMSE 2.995586
#> RMSE...5 Fold5.Rep01  RMSE 3.739492
#> RMSE...6 Fold1.Rep02  RMSE 3.931772
```

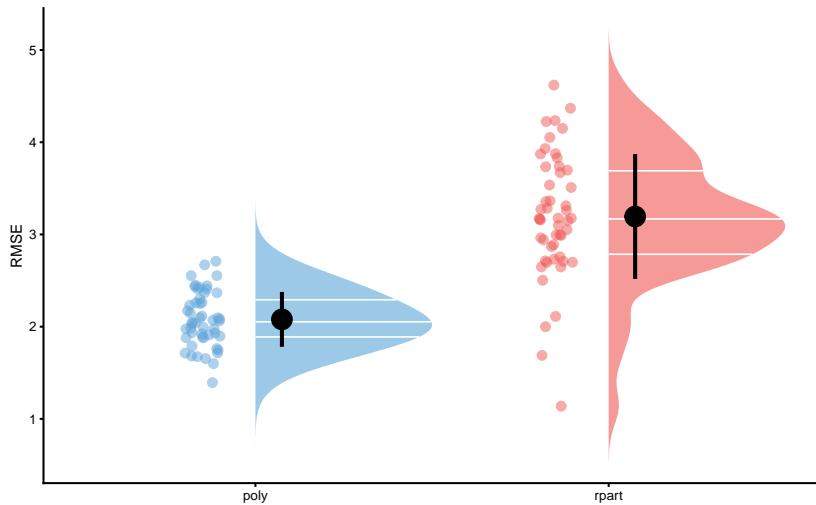
```

rmse_poly <- filter(model4$cross_validation$performance$folds$testing,
                      metric == "RMSE")
head(rmse_poly)
#>           fold metric    value
#> RMSE...1 Fold1.Rep01  RMSE 1.745409
#> RMSE...2 Fold2.Rep01  RMSE 2.408312
#> RMSE...3 Fold3.Rep01  RMSE 2.039587
#> RMSE...4 Fold4.Rep01  RMSE 2.670888
#> RMSE...5 Fold5.Rep01  RMSE 2.070925
#> RMSE...6 Fold1.Rep02  RMSE 2.082951

rmse_data <- rbind(data.frame(model = "rpart", value = rmse_rpart$value),
                     data.frame(model = "poly", value = rmse_poly$value))

bmbstats::plot_raincloud(rmse_data, value = "value", value_label = "RMSE",
                          groups = "model")

```



And we can finally perform the dependent groups analysis:

```

rmse_perf <- bmbstats::compare_dependent_groups(pre = rmse_rpart$value,
                                                 post = rmse_poly$value)

rmse_perf
#> Bootstrap with 2000 resamples and 95% bca confidence intervals.
#>
#>           estimator      value      lower
#> SESOI lower -0.13539863 -0.17607368

```

```

#>      SESOI upper   0.13539863   0.10927225
#>      SESOI range   0.27079726   0.21854450
#>      Mean change -1.11530929 -1.32646984
#>      SD change   0.76022031   0.60218574
#>      %CV change -68.16228649 -108.17299822
#>      % change    -30.38499807 -35.93194876
#>      Ratio       0.69615002   0.64068051
#>      Cohen's d   -1.64744544 -2.17427538
#>      CLES        0.06573315   0.02725296
#>      OVL         0.41009713   0.27810102
#>      Mean change to SESOI -4.11861361 -5.43568845
#>      SD change to SESOI  2.80734121  2.51010263
#>      pLower      0.89827180   0.76960554
#>      pEquivalent 0.04856260   0.02183432
#>      pHigher     0.05316560   0.01748864
#>      upper
#>      -0.10927225
#>      0.17607368
#>      0.35214735
#>      -0.89455705
#>      0.99219147
#>      -50.50157994
#>      -19.10373522
#>      0.80896265
#>      -1.02403852
#>      0.16669235
#>      0.60868406
#>      -2.56009630
#>      3.14489890
#>      0.95994450
#>      0.09330646
#>      0.13895348

```

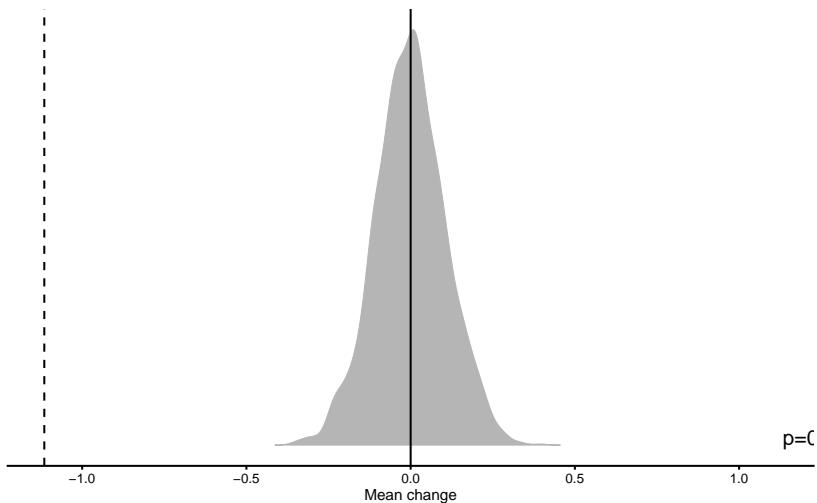
```

# Do the statistical significance test
rmse_NHST <- bmbstats::bootstrap_NHST(rmse_perf, estimator = "Mean change",
                                         test = "two.sided", null_hypothesis = 0)

rmse_NHST
#> Null-hypothesis significance test for the `Mean change` estimator
#> Bootstrap result: Mean change=-1.115, 95% CI [-1.326, -0.895]
#> H0=0, test: two.sided
#> p=0.000499750124937531

```

```
plot(rmse_NHST)
```

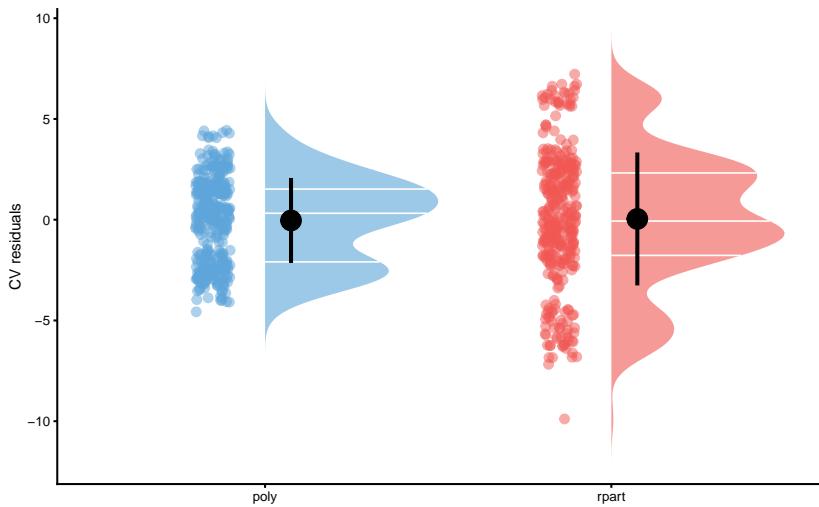


So we can conclude that polynomial model has significantly better performance, using RMSE estimator, than Regression Tree model that cannot be explained as a pure (sampling) chance.

But to be honest, model comparison is beyond my current knowledge and I am not sure that comparing the estimators is the right approach. We could instead compare model residuals. Since CV folds are identical, we can perform dependent groups analysis as well. But before doing that, let's plot the CV residuals:

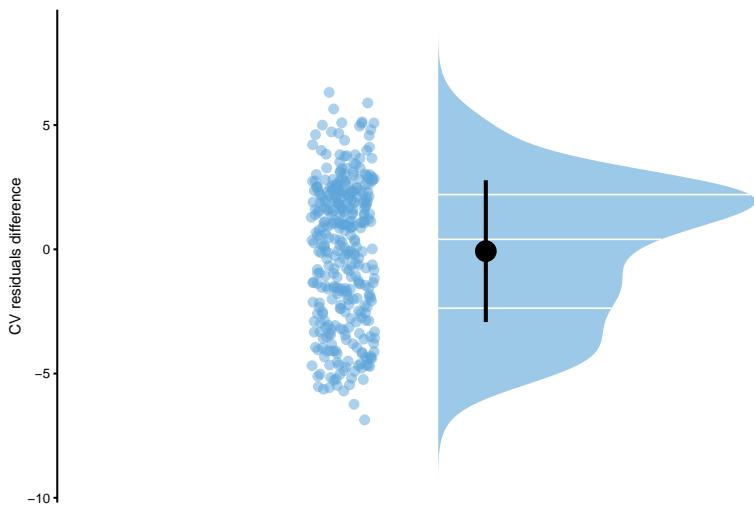
```
resid_data <- rbind(data.frame(model = "rpart", value = model3$cross_validation$data$testing$residual),
                      data.frame(model = "poly", value = model4$cross_validation$data$testing$residual))

bmbstats::plot_raincloud(resid_data, value = "value", value_label = "CV residuals",
                         groups = "model")
```



Or their difference:

```
bmbstats::plot_raincloud(data.frame(diff = model4$cross_validation$data$testing$residual -
model3$cross_validation$data$testing$residual), value = "diff",
value_label = "CV residuals difference")
```



And we can finally perform the dependent groups analysis:

```
resid_perf <- bmbstats::compare_dependent_groups(pre = model3$cross_validation$data$testing$residual,
post = model4$cross_validation$data$testing$residual)
```

```

resid_perf
#> Bootstrap with 2000 resamples and 95% bca confidence intervals.
#>
#>      estimator      value      lower
#> SESOI lower -6.596246e-01 -7.094456e-01
#> SESOI upper  6.596246e-01  6.153383e-01
#> SESOI range   1.319249e+00  1.230677e+00
#> Mean change -7.485741e-02 -3.678773e-01
#> SD change   2.853680e+00  2.711641e+00
#> %CV change -3.812154e+03 -1.203382e+06
#> % change    -1.816303e+01 -8.898433e+01
#> Ratio        8.183697e-01  1.101567e-01
#> Cohen's d   -2.269697e-02 -1.118320e-01
#> CLES          4.923722e-01  4.625656e-01
#> OVL           9.909454e-01  9.689477e-01
#> Mean change to SESOI -5.674243e-02 -2.795801e-01
#> SD change to SESOI  2.163109e+00  2.042661e+00
#> pLower        4.188782e-01  3.767829e-01
#> pEquivalent   1.826035e-01  1.721664e-01
#> pHIGHER       3.985183e-01  3.604887e-01
#> upper
#> -6.153383e-01
#> 7.094456e-01
#> 1.418891e+00
#> 2.267774e-01
#> 3.019229e+00
#> -1.112797e+03
#> 6.621511e+01
#> 1.662151e+00
#> 6.659056e-02
#> 5.225228e-01
#> 9.999341e-01
#> 1.664764e-01
#> 2.298416e+00
#> 4.605721e-01
#> 1.933616e-01
#> 4.386698e-01

```

```

# Do the statistical significance test
resid_NHST <- bmbstats::bootstrap_NHST(resid_perf, estimator = "Mean change",
                                         test = "two.sided", null_hypothesis = 0)

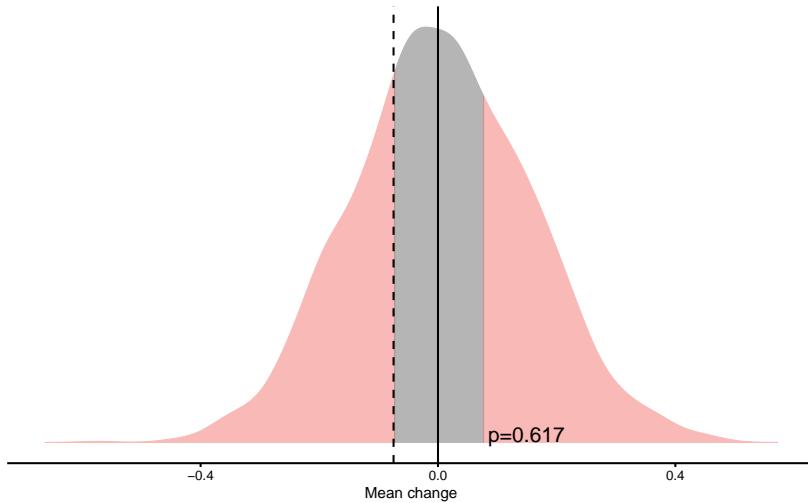
resid_NHST
#> Null-hypothesis significance test for the `Mean change` estimator

```

```
#> Bootstrap result: Mean change=-0.075, 95% CI [-0.368, 0.227]
#> H0=0, test: two.sided
```

```
#> p=0.617
```

```
plot(resid_NHST)
```



According to Cross-Validation residuals analysis, the two models didn't perform statistically different than can be attributed to chance.

It bears repeating that I am not sure either of these are valid model comparison methods, so use this only as an example. Model comparison also involves deciding about model complexity and selecting the simpler model.

14.6 Bootstrapping model

Cross-validations is one technique that allows us to evaluate model performance on unseen data, check for model over-fitting, and finally to tune the model parameters (i.e. using tuning parameters, like we did by varying the polynomial degrees) to get the best predictive performance.

With statistical inferences we are interested in *generalizing* beyond our sample, to a bigger population from which the sample is taken. We can generalize model parameters (i.e. intercept and slope of parameter coefficients), but also model performance (i.e. what is the *true* relationship between two or more variables). We can thus bootstrap the above model to get confidence intervals for both parameters and performance estimators.

Let's get the 95% bootstrapped CIs for the polynomial model:

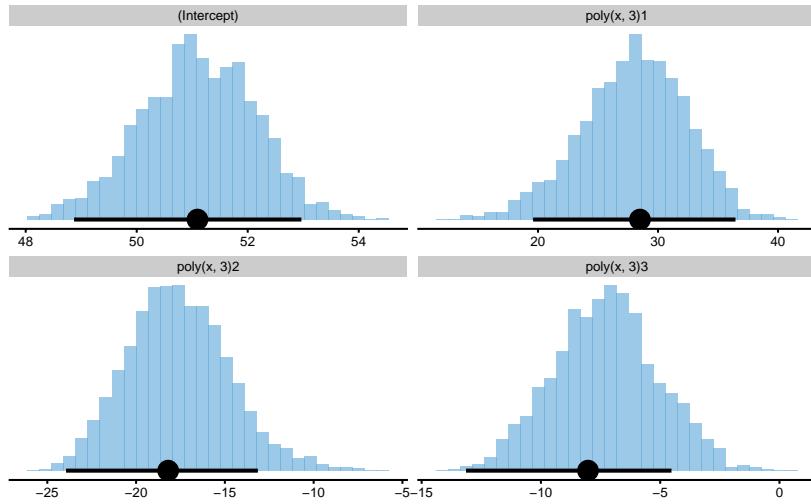
```

model4.boot.coef <- bmbstats::bmbstats(data = sinus_data,
  estimator_function = function(data, SESOI_lower, SESOI_upper,
    na.rm, init_boot) {
  model <- lm(observed_y ~ poly(x, 3), data)
  coef(model)
})

model4.boot.coef
#> Bootstrap with 2000 resamples and 95% bca confidence intervals.
#>
#>   estimator      value     lower      upper
#>   (Intercept)  51.094279  48.87153  52.96739
#>   poly(x, 3)1  28.497988  19.59403  36.45944
#>   poly(x, 3)2 -18.194924 -23.94497 -13.14120
#>   poly(x, 3)3 -8.027118 -13.13937 -4.52746

```

```
plot(model4.boot.coef)
```



And here are the model performance estimators and their 95% bootstrap confidence intervals:

```

model4.boot.perf <- bmbstats::bmbstats(data = sinus_data,
  estimator_function = function(data, SESOI_lower, SESOI_upper,
    na.rm, init_boot) {
  model <- lm(observed_y ~ poly(x, 3), data)

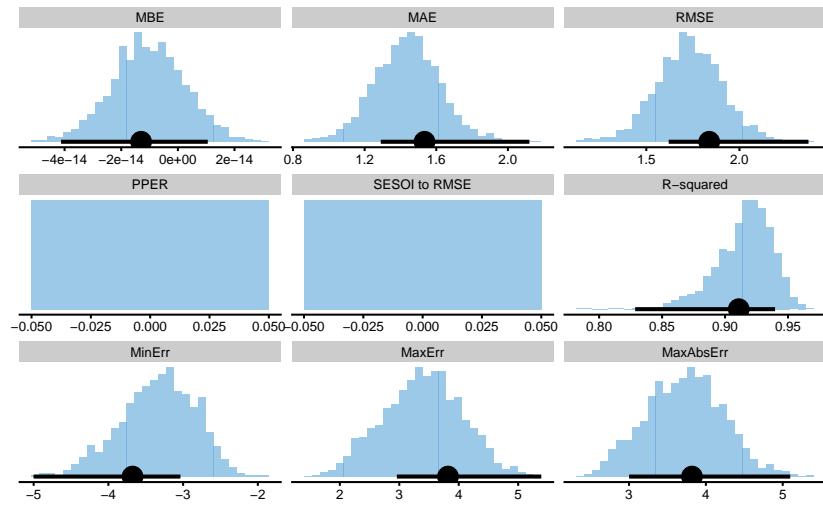
  # Return model performance metrics

```

```
bmbstats::performance_metrics(observed = data$observed_y,
                               predicted = predict(model), SESOI_lower = SESOI_lower,
                               SESOI_upper = SESOI_upper, na.rm = na.rm)
  })

model4.boot.perf
#> Bootstrap with 2000 resamples and 95% bca confidence intervals.
#>
#>      estimator       value      lower
#>      MBE -1.299275e-14 -4.125793e-14
#>      MAE  1.534398e+00  1.290075e+00
#>      RMSE 1.837539e+00  1.620042e+00
#>      PPER  0.000000e+00          NA
#> SESOI to RMSE 0.000000e+00          NA
#>      R-squared 9.108623e-01  8.286065e-01
#>      MinErr -3.676547e+00 -5.001168e+00
#>      MaxErr  3.819068e+00  2.958894e+00
#>      MaxAbsErr 3.819068e+00  3.001014e+00
#>      upper
#>      1.055661e-14
#>      2.119985e+00
#>      2.370531e+00
#>      NA
#>      NA
#>      9.396944e-01
#>      -3.035433e+00
#>      5.388544e+00
#>      5.097689e+00

plot(model4.boot.perf)
```



It is important to remember that these model analysis are trying to answer different questions - one predictive and one inferential.

Chapter 15

Validity and Reliability

Although we have *sort-of-touched* validity and reliability topics in the [Measurement Error](#) chapter, here I will spend more time explaining the concepts and the use of `bmbstats` functions to perform the analysis. As pointed out multiple times thorough this book, simulation and the knowledge of the data generating process (DGP) is very helpful in understanding the analysis and what we are trying to do. To understand validity and reliability analyses, it is thus best to generate the data (i.e. simulate the DGP).

15.1 Data generation

Let's imagine that we know the *true* vertical jump scores for N=20 individuals. We measure this true score with a *gold standard* device twice (i.e. trial 1 and trial 2). Unfortunately, this *criterion* device is expensive or takes a lot of time, so we have developed a *practical* device that measures vertical jump height much quicker and cheaper. Unfortunately, we do not know the measurement error of this device and we want to estimate how valid and reliable this practical measure is. Luckily for us, we generate the data, so we actually know all the measurement error components (i.e. fixed and proportional bias components of the systematic error and random error).

The below code generates the data for both criterion and practical measures using two trials and N=20 athletes:

```
require(tidyverse)
require(bmbstats)
require(cowplot)

n_subjects <- 20
```

```

criterion_random <- 0.3
practical_fixed <- 2
practical_proportional <- 1.1
practical_random <- 1

set.seed(1667)

agreement_data <- tibble(Athlete = paste("Athlete", str_pad(string = seq(1,
  n_subjects), width = 2, pad = "0")), True_score = rnorm(n_subjects,
  45, 5), Criterion_score.trial1 = 0 + (True_score * 1) +
  rnorm(n_subjects, 0, criterion_random), Criterion_score.trial2 = 0 +
  (True_score * 1) + rnorm(n_subjects, 0, criterion_random),
  Practical_score.trial1 = practical_fixed + (True_score *
  practical_proportional) + rnorm(n_subjects, 0, practical_random),
  Practical_score.trial2 = practical_fixed + (True_score *
  practical_proportional) + rnorm(n_subjects, 0, practical_random))

head(agreement_data)
#> # A tibble: 6 x 6
#>   Athlete  True_score Criterion_score~ Criterion_score~
#>   <chr>     <dbl>          <dbl>          <dbl>
#> 1 Athlet~    52.9           52.9           52.9
#> 2 Athlet~    42.4           42.3           42.2
#> 3 Athlet~    49.2           49.1           49.5
#> 4 Athlet~    44.8           44.7           44.8
#> 5 Athlet~    40.0           40.4           40.1
#> 6 Athlet~    42.6           42.5           42.2
#> # ... with 2 more variables:
#> #   Practical_score.trial1 <dbl>,
#> #   Practical_score.trial2 <dbl>

```

The assumption of the above DGP is that true score stays unchanged for trial 1 and trial 2. Thus, the only thing that creates variance in the criterion and practical measures is the random error component of the measurement error.

It is also assumed that there is no *biological noise* involved in the measurement error. We will make this example a bit more complex later on, so for now let's stick to this sand box example.

15.2 Validity

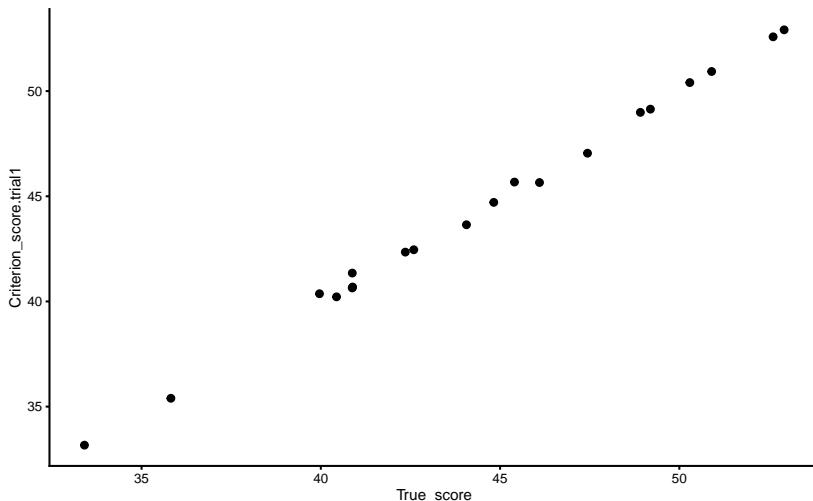
“How well does the measure measure what it’s supposed to measure?” (Will G Hopkins 2004b; Hopkins 2010; W. Hopkins 2015) is the question validity

analysis tries to answer. There are few common approaches to estimate validity of the measurements, with Bland-Altman analysis (Bland and Altman 1986; Giavarina 2015) being one of the most used (also referred to as the *method of the differences*), linear regression, and *ordinary least products* (OLP) being a better alternative (Ludbrook 2010, 2012, 1997, 2002; Mullineaux, Barnes, and Batterham 1999). Since we have simulated the DGP, we know exactly the measurement error components as well as true scores (which we do not know in the *real life*).

15.2.1 True vs Criterion

To *re-create* DGP parameters for the criterion measure, we will use the true scores as a predictor, and criterion score (trial 1) as the outcome. Here is the scatter plot:

```
ggplot(agreement_data, aes(x = True_score, y = Criterion_score.trial1)) +
  theme_cowplot(8) + geom_point()
```

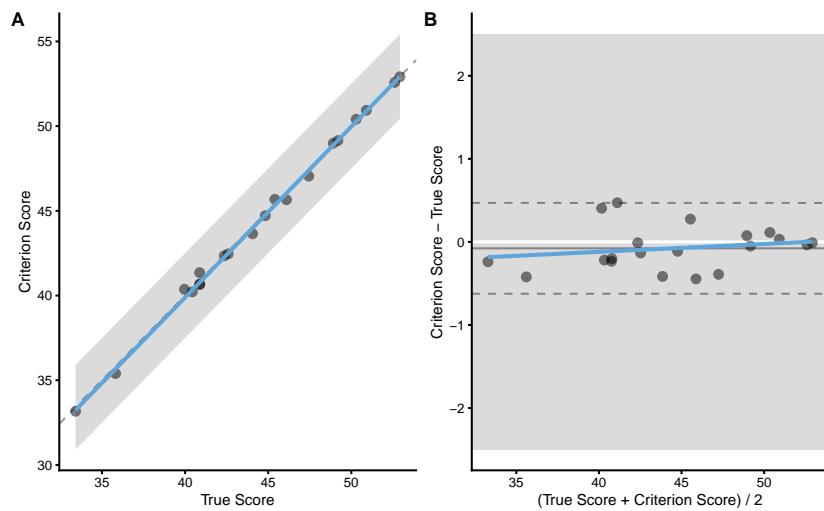


For the sake of this example, let's assume that SESOI is equal to $\pm 2.5\text{cm}$ (for the criterion score). We are interested in few things: (1) re-create the DGP components, or (2) can we predict the true score from the criterion score (which implies flipping predictor and outcome variables). Most validity research papers in the sports science are concerned with describing or explaining the validity by trying to re-create the DGP components, while not many are concerned with predictive performance of the model. Let's deal with the descriptive (i.e explanatory) tasks first.

15.2.1.1 Method of the differences

Using the `bmbstats::plot_pair_BA` function we can generate Bland-Altman plot for the true score and the criterion score:

```
bmbstats::plot_pair_BA(predictor = agreement_data$True_score,
  outcome = agreement_data$Criterion_score.trial1, predictor_label = "True Score",
  outcome_label = "Criterion Score", SESOI_lower = -2.5,
  SESOI_upper = 2.5)
```



Panel A in the previous figure depicts simple scatter plot with added *identity line* (dashed line), SEMOI band around identity line, and linear regression model (blue line; this could be changed using the `control = bmbstats::plot_contrn()` parameter). Panel B depicts the difference between criterion and true score (oy y-axis) and their average (on the x-axis). Using SEMOI of ± 2.5 cm, we can conclude that all the differences fall within the SEMOI band, confirming that the criterion measure has outstanding practical validity characteristic.

How do we re-create the DGP parameters? Well, using Bland-Altman method, both fixed and proportional bias are lumped together in the `bias` (i.e. `mean difference`) estimator, and random error component is estimated using `SD` of the differences. Let's write a simple estimator function to perform method of the differences validity analysis (do not mind the names of the function parameters; I know they make thing more confusing, but bear with me for a second):

```
differences_method <- function(data, criterion, practical,
  SESOI_lower = 0, SESOI_upper = 0, na.rm = FALSE) {
  practical_obs <- data[[practical]]
```

```

criterion_obs <- data[[criterion]]

SESOI_range <- SESOI_upper - SESOI_lower

diff <- criterion_obs - practical_obs

n_obs <- length(diff)

mean_diff <- mean(diff, na.rm = na.rm)
sd_diff <- sd(diff, na.rm = na.rm)

PPER <- stats::pt((SESOI_upper - mean_diff)/sd_diff,
                   df = n_obs - 1) - stats::pt((SESOI_lower - mean_diff)/sd_diff,
                   df = n_obs - 1)

c(`Mean diff` = mean_diff, `SD diff` = sd_diff, PPER = PPER)
}

# Run the analysis
differences_method(data = agreement_data, criterion = "Criterion_score.trial1",
                     practical = "True_score", SESOI_lower = -2.5, SESOI_upper = 2.5)
#>   Mean diff      SD diff      PPER
#> -0.07726199  0.26125825  0.99999999

```

We do know that the random error for the criterion score is 0.3cm since we have generated the data, and SD diff is our estimate of that parameter. Let's perform bootstrap method to get confidence intervals for these estimators using `bmbstats::validity_analysis` function with `differences_method` as a parameter:

```

difference_validity <- bmbstats::validity_analysis(data = agreement_data,
                                                    criterion = "Criterion_score.trial1", practical = "True_score",
                                                    SESOI_lower = -2.5, SESOI_upper = 2.5, estimator_function = differences_method,
                                                    control = model_control(seed = 1667))

difference_validity
#> Bootstrap with 2000 resamples and 95% bca confidence intervals.
#>
#>   estimator      value      lower      upper
#>   Mean diff -0.07726199 -0.1768739  0.04312759
#>   SD diff   0.26125825  0.1964889  0.34579624
#>   PPER      0.99999999  0.9999993  1.00000000

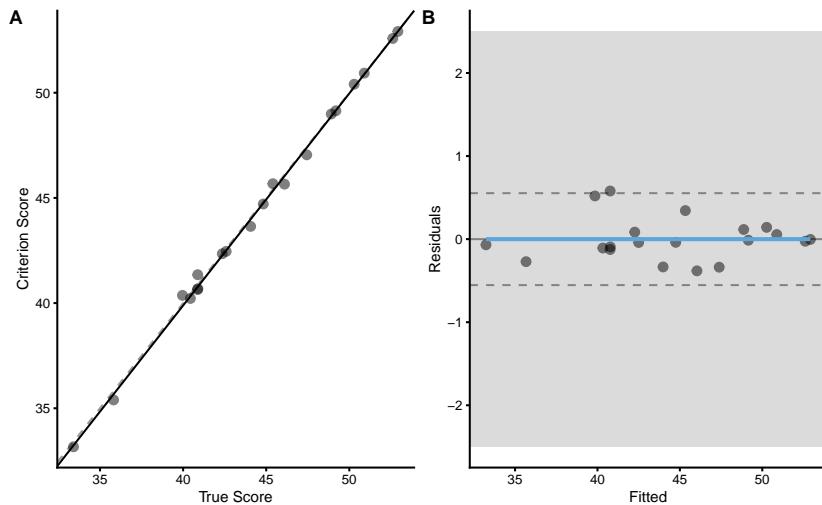
```

As can be seen in the results, 95% CIs for the SD diff captures the true DGP parameter value for the random error of the criterion measure.

15.2.1.2 Linear Regression method

Another approach involves using simple linear regression method, where RSE is used to estimate PPER estimator. Using linear regression we can estimate the intercept (i.e. fixed bias), slope (i.e. proportional bias), and random error (i.e. RSE). Before writing the estimators function, let's plot the relationship using `bmbstats::plot_pair_lm`:

```
bmbstats::plot_pair_lm(predictor = agreement_data$True_score,
                      outcome = agreement_data$Criterion_score.trial1, predictor_label = "True Score",
                      outcome_label = "Criterion Score", SESOI_lower = -2.5,
                      SESOI_upper = 2.5)
```



The linear regression parameters are used as estimates of the measurement error:

```
lm_method <- function(data, criterion, practical, SESOI_lower = 0,
                      SESOI_upper = 0, na.rm = FALSE) {
  practical_obs <- data[[practical]]
  criterion_obs <- data[[criterion]]

  SESOI_range <- SESOI_upper - SESOI_lower

  lm_model <- lm(criterion_obs ~ practical_obs)

  n_obs <- length(criterion_obs)

  intercept <- coef(lm_model)[[1]]
  slope <- coef(lm_model)[[2]]
```

```
rse <- summary(lm_model)$sigma

# This is very close to 0, but will use it nonetheless
mean_diff <- mean(residuals(lm_model))

PPER <- stats::pt((SESOI_upper - mean_diff)/rse, df = n_obs -
  1) - stats::pt((SESOI_lower - mean_diff)/rse, df = n_obs -
  1)

c(Intercept = intercept, Slope = slope, RSE = rse, PPER = PPER)
}

# Run the analysis
lm_method(data = agreement_data, criterion = "Criterion_score.trial1",
  practical = "True_score", SESOI_lower = -2.5, SESOI_upper = 2.5)
#> Intercept      Slope      RSE      PPER
#> -0.4518909  1.0084198  0.2643645  1.0000000
```

We will use `bmbstats::validity_analysis` to get 95% CIs around the estimates:

```
lm_validity <- bmbstats::validity_analysis(data = agreement_data,
  criterion = "Criterion_score.trial1", practical = "True_score",
  SESOI_lower = -2.5, SESOI_upper = 2.5, estimator_function = lm_method,
  control = model_control(seed = 1667))

lm_validity
#> Bootstrap with 2000 resamples and 95% bca confidence intervals.
#>
#>   estimator      value      lower      upper
#>   Intercept -0.4518909 -1.0759400  0.8336904
#>   Slope     1.0084198  0.9825368  1.0220937
#>   RSE       0.2643645  0.1859689  0.3575745
#>   PPER      1.0000000  0.9999988  1.0000000
```

Since we know the DGP behind the criterion measure, we can confirm that fixed bias estimate (i.e. intercept) CI captured 0, proportional bias (i.e. slope) captured 1 and that random error (i.e. RSE) captured 0.3. We have thus managed to re-create DGP parameters using simple linear regression.

We can also perform NHST for fixed and proportional bias, and random error (RSE):

```
bmbstats::bootstrap_NHST(lm_validity, estimator = "Intercept",
  null_hypothesis = 0, test = "two.sided")
#> Null-hypothesis significance test for the `Intercept` estimator
#> Bootstrap result: Intercept=-0.452, 95% CI [-1.076, 0.834]
#> H0=0, test: two.sided
#> p=0.262

bmbstats::bootstrap_NHST(lm_validity, estimator = "Slope",
  null_hypothesis = 1, test = "two.sided")
#> Null-hypothesis significance test for the `Slope` estimator
#> Bootstrap result: Slope=1.008, 95% CI [0.983, 1.022]
#> H0=1, test: two.sided
#> p=0.322

bmbstats::bootstrap_NHST(lm_validity, estimator = "RSE",
  null_hypothesis = criterion_random, test = "two.sided")
#> Null-hypothesis significance test for the `RSE` estimator
#> Bootstrap result: RSE=0.264, 95% CI [0.186, 0.358]
#> H0=0.3, test: two.sided
#> p=0.427
```

Please notice the similarity between `SD diff` from the method of differences and `RSE` estimator, both of which are estimates of the random error of the measurement error:

```
rbind(difference_validity$estimators[2, ], lm_validity$estimators[3,
  ])
#>   estimator      value    lower     upper
#> 2   SD diff  0.2612583 0.1964889 0.3457962
#> 3       RSE  0.2643645 0.1859689 0.3575745
```

15.2.1.3 OLP method

Ordinary least product approach calculates the residuals using product between y-residual $y - \hat{y}$ and x-residual $x - \hat{x}$. This method is used more in reliability analysis when we do not know which variable is predictor and which is the outcome or target, but it can be used with the validity analysis as well. OLP method doesn't assume that the x-variable is without random error, which in this example doesn't help since we are using the true score (and the true score is without random error). But it is useful *exactly* for this reason for the real-world analysis when both variables (x and y) have random error involved. Here is a quote from Ludbrook 1997 paper (Ludbrook 1997, 194):

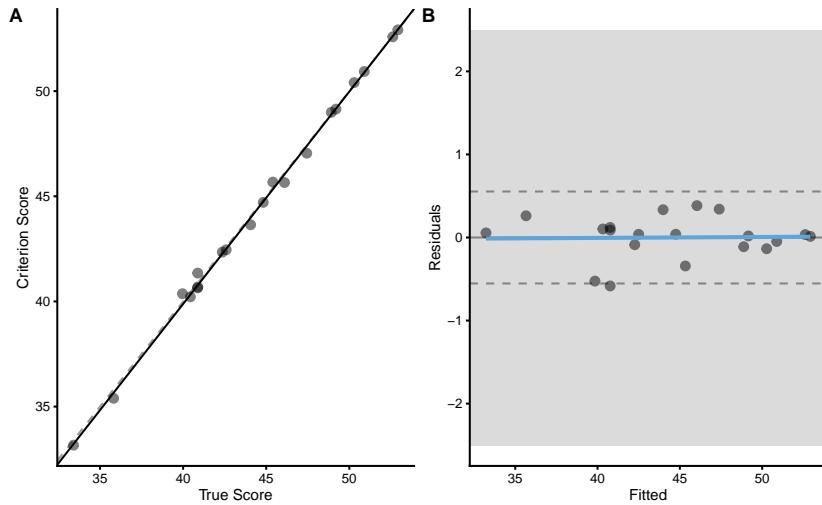
"It is an important assumption of OLS regression that whereas the values of Y in the population that has been sampled are attended by error, those of X are not. Strictly, this can be so only if the X values are categorical: for instance, conditions, treatments or places. However, most statistical theorists follow the advice of Berkson, which is that if the X values have been fixed in advance by the experimenter (e.g. by specifying times, doses or settings of a pump), then they can be regarded for practical purposes as error-free. When X is error-free, Model I regression analysis is the proper form to use. It includes the well-known OLS regression technique as well as modifications of it, such as weighted least squares (WLS) regression.

When both X and Y are free to vary and are attended by error, some statisticians allow that Model I regression analysis may still be used if it is certain, on biological grounds, that Y must depend on X and never the reverse. This is the case, for instance, in dose- or stimulus-response relationships. Even then, the Model I regression line should be used for empirical, rather than explanatory, purposes.

When investigators plan experiments to compare methods of measurement, they must assume that both Y and X will be attended by random error. Moreover, it is impossible to decide which method should be regarded as dependent and which independent and because of this it is wrong to use Model I regression analysis. Instead, one or another form of Model I regression analysis must be used. These are described later."

To plot OLP regression, use `bmbstats::plot_pair_OLP` function. The OLP regression, in this case, looks very close to simple linear regression:

```
bmbstats::plot_pair_OLP(predictor = agreement_data$True_score,
outcome = agreement_data$Criterion_score.trial1, predictor_label = "True Score",
outcome_label = "Criterion Score", SESOI_lower = -2.5,
SESOI_upper = 2.5)
```



Let's write the OLP validity estimators and check the output. OLP regression is implemented in the `bmbstats::OLP_regression` function.

```
olp_method <- function(data, criterion, practical, SESOI_lower = 0,
  SESOI_upper = 0, na.rm = FALSE) {
  practical_obs <- data[[practical]]
  criterion_obs <- data[[criterion]]

  SESOI_range <- SESOI_upper - SESOI_lower

  olp_model <- bmbstats::OLP_regression(outcome = criterion_obs,
    predictor = practical_obs, na.rm = na.rm)

  n_obs <- length(criterion_obs)

  intercept <- olp_model$intercept
  slope <- olp_model$slope
  rse <- olp_model$rse

  PPER <- stats::pt((SESOI_upper)/rse, df = n_obs - 1) -
    stats::pt((SESOI_lower)/rse, df = n_obs - 1)

  c(Intercept = intercept, Slope = slope, RSE = rse, PPER = PPER)
}

olp_method(data = agreement_data, criterion = "Criterion_score.trial1",
  practical = "True_score", SESOI_lower = -2.5, SESOI_upper = 2.5)
```

```
#> Intercept      Slope       RSE      PPER
#> -0.5024828  1.0095568  0.2644390  1.0000000
```

To estimate 95% CIs for these estimators, use the *default* `bmbstats::validity_analysis`:

```
olp_validity <- bmbstats::validity_analysis(data = agreement_data,
criterion = "Criterion_score.trial1", practical = "True_score",
SESOI_lower = -2.5, SESOI_upper = 2.5, estimator_function = olp_method,
control = model_control(seed = 1667))

olp_validity
#> Bootstrap with 2000 resamples and 95% bca confidence intervals.
#>
#>   estimator      value     lower      upper
#>   Intercept -0.5024828 -1.1085643  0.7298272
#>   Slope     1.0095568  0.9843866  1.0226211
#>   RSE       0.2644390  0.1859924  0.3579299
#>   PPER     1.0000000  0.9999988  1.0000000
```

15.2.1.4 What happens if we flip the variables?

In the DGP we have used true scores to generate both criterion and practical measures, thus using true score as predictor (x-variable) and criterion as the outcome (y-variable) is valid approach to re-create the DGP parameters. Please note that `bmbstats::validity_analysis` uses `practical` parameter as predictor (x-variable) and `criterion` parameter as outcome variable (y-variable).

But in the real life (as we will soon see when using criterion and practical measures) we do not know the true scores and we might be interested in *predicting* criterion from the practical measures. Let's see what happens when we flip the predictor and outcome for all three methods:

```
difference_validity_flip <- bmbstats::validity_analysis(data = agreement_data,
practical = "Criterion_score.trial1", criterion = "True_score",
SESOI_lower = -2.5, SESOI_upper = 2.5, estimator_function = differences_method,
control = model_control(seed = 1667))

difference_validity_flip
#> Bootstrap with 2000 resamples and 95% bca confidence intervals.
#>
#>   estimator      value     lower      upper
#>   Mean diff  0.07726199 -0.04312759  0.1768739
#>   SD diff   0.26125825  0.19648887  0.3457962
#>   PPER      0.99999999  0.99999933  1.0000000
```

```

lm_validity_flip <- bmbstats::validity_analysis(data = agreement_data,
  practical = "Criterion_score.trial1", criterion = "True_score",
  SESOI_lower = -2.5, SESOI_upper = 2.5, estimator_function = lm_method,
  control = model_control(seed = 1667))

lm_validity_flip
#> Bootstrap with 2000 resamples and 95% bca confidence intervals.
#>
#>   estimator      value     lower      upper
#>   Intercept 0.5472786 -0.6599661 1.1190949
#>   Slope     0.9894180  0.9770752 1.0135275
#>   RSE       0.2618619  0.1813661 0.3559808
#>   PPER     1.0000000  0.9999989 1.0000000

olp_validity_flip <- bmbstats::validity_analysis(data = agreement_data,
  practical = "Criterion_score.trial1", criterion = "True_score",
  SESOI_lower = -2.5, SESOI_upper = 2.5, estimator_function = olp_method,
  control = model_control(seed = 1667))

olp_validity_flip
#> Bootstrap with 2000 resamples and 95% bca confidence intervals.
#>
#>   estimator      value     lower      upper
#>   Intercept 0.4977261 -0.7419619 1.0837733
#>   Slope     0.9905337  0.9778841 1.0158849
#>   RSE       0.2619357  0.1812142 0.3560398
#>   PPER     1.0000000  0.9999989 1.0000000

```

Let's combine the result for the random error estimates so we can compare them all:

```

rand_err_estimates <- rbind(data.frame(outcome = "Criterion",
  method = "Difference", difference_validity$estimators[2,
    ]),
  data.frame(outcome = "True", method = "Difference",
  difference_validity_flip$estimators[2, ]), data.frame(outcome = "Criterion",
  method = "lm", lm_validity$estimators[3, ]), data.frame(outcome = "True",
  method = "lm", lm_validity_flip$estimators[3, ]), data.frame(outcome = "Criterion",
  method = "olp", olp_validity$estimators[3, ]), data.frame(outcome = "True",
  method = "olp", olp_validity_flip$estimators[3, ]))

print(rand_err_estimates, row.names = FALSE)
#>   outcome      method estimator      value     lower
#>   Criterion Difference      SD diff 0.2612583 0.1964889
#>   True Difference      SD diff 0.2612583 0.1964889
#>   Criterion        lm      RSE 0.2643645 0.1859689

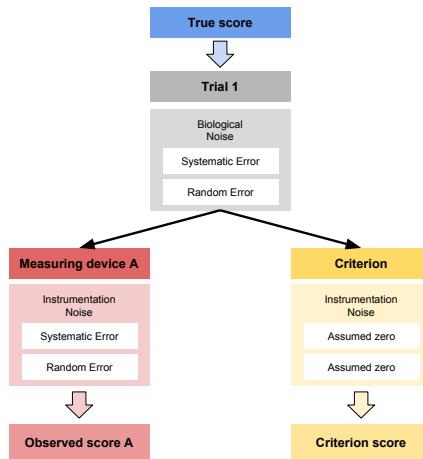
```

```
#>      True      lm      RSE 0.2618619 0.1813661
#> Criterion olp      RSE 0.2644390 0.1859924
#>      True      olp      RSE 0.2619357 0.1812142
#>      upper
#> 0.3457962
#> 0.3457962
#> 0.3575745
#> 0.3559808
#> 0.3579299
#> 0.3560398
```

As can be seen from the table, the estimates for the random error component of the criterion measure are very similar (although `SD diff` is smaller due dividing by $N - 1$, while `RSE` uses $N - k - 1$, where k is number of predictors, thus we get $N - 2$) regardless of the method and which variable is the predictor and which is outcome. But this is the case since true score is the *true score* without measurement error. Let's see what happens in the *real-world*.

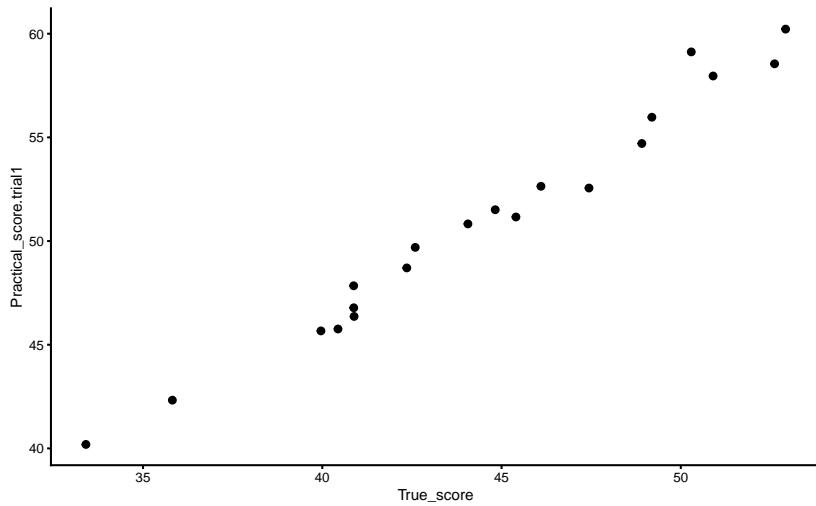
15.2.2 Practical vs Criterion

In the real-world, we do not know the true scores. We can only use some type of the gold standard measure. In our DGP simulation we have generated both criterion and practical measures using the known true scores and measurement error for the each measure.



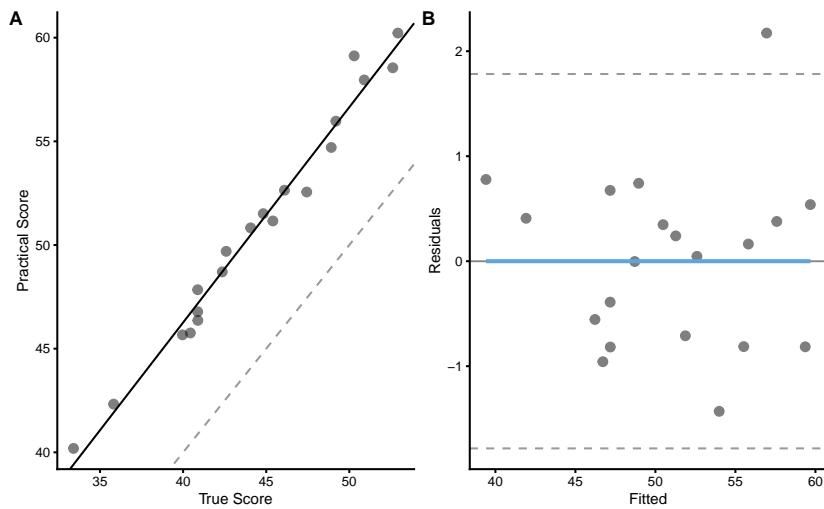
Let's check if we can re-create DGP parameters for the practical measure (which has both proportional and fixed bias, as well as larger random error than criterion) using the criterion score. Here is the scatter plot of the two:

```
ggplot(agreement_data, aes(x = True_score, y = Practical_score.trial1)) +
  theme_cowplot(8) + geom_point()
```



To re-create DGP for the practical measure, we need to use true score as the predictor (since that is how we have generated the practical scores). Let's use the simple linear regression method to do so:

```
bmbstats::plot_pair_lm(predictor = agreement_data$True_score,
  outcome = agreement_data$Practical_score.trial1, predictor_label = "True Score",
  outcome_label = "Practical Score")
```



And let's estimate the 95% bootstrap confidence intervals:

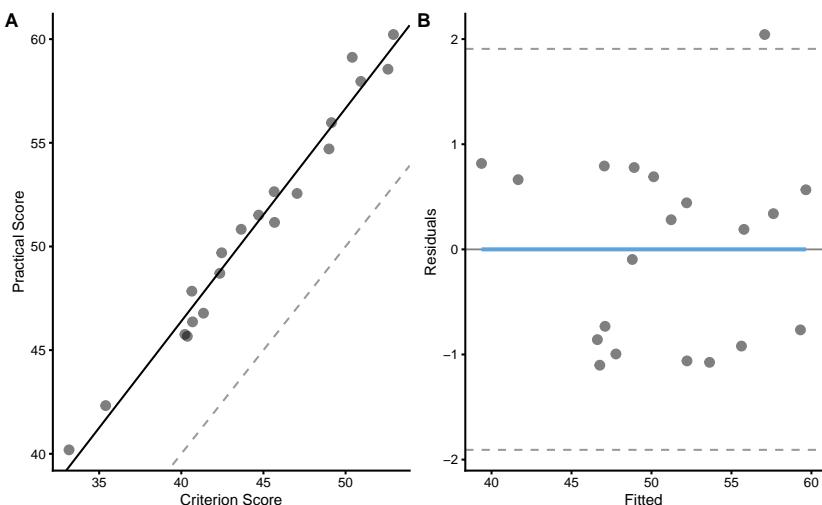
```
lm_validity_practical <- bmbstats::validity_analysis(data = agreement_data,
  criterion = "Practical_score.trial1", practical = "True_score",
  SESOI_lower = NA, SESOI_upper = NA, estimator_function = lm_method,
  control = model_control(seed = 1667))

lm_validity_practical
#> Bootstrap with 2000 resamples and 95% bca confidence intervals.
#>
#>   estimator      value    lower     upper
#>   Intercept 4.7157542 0.6929173 7.345273
#>   Slope     1.0386315 0.9758422 1.132740
#>   RSE       0.8518886 0.6168049 1.304054
#>   PPER        NA        NA        NA
```

In our DGP, we have used fixed bias equal to 2cm, proportional bias equal to 1.1, and random error equal to 1cm to generate the practical scores. As can be seen from results, our 95% CI captured the true DGP parameter values.

Unfortunately, we have to use the criterion measure, since we do not know the true scores. Let's check the results:

```
bmbstats::plot_pair_lm(predictor = agreement_data$Criterion_score.trial1,
  outcome = agreement_data$Practical_score.trial1, predictor_label = "Criterion Score",
  outcome_label = "Practical Score")
```



And let's estimate the bootstrap 95% CIs:

```
lm_validity_practical_criterion <- bmbstats::validity_analysis(data = agreement_data,
  criterion = "Practical_score.trial1", practical = "Criterion_score.trial1",
  SESOI_lower = NA, SESOI_upper = NA, estimator_function = lm_method,
  control = model_control(seed = 1667))

lm_validity_practical_criterion
#> Bootstrap with 2000 resamples and 95% bca confidence intervals.
#>
#>   estimator      value     lower      upper
#>   Intercept 5.3054383 0.8089323 8.054704
#>       Slope 1.0271620 0.9616332 1.126357
#>       RSE 0.9111659 0.7303417 1.224814
#>       PPER        NA        NA        NA
```

Since criterion measure (in this case used as predictor, or x-variable) also contains random error, OLP method can be used instead:

```
olp_validity_practical_criterion <- bmbstats::validity_analysis(data = agreement_data,
  criterion = "Practical_score.trial1", practical = "Criterion_score.trial1",
  SESOI_lower = NA, SESOI_upper = NA, estimator_function = olp_method,
  control = model_control(seed = 1667))

olp_validity_practical_criterion
#> Bootstrap with 2000 resamples and 95% bca confidence intervals.
#>
#>   estimator      value     lower      upper
#>   Intercept 4.7308268 -0.1090732 7.659421
#>       Slope 1.0400989 0.9737440 1.150441
#>       RSE 0.9140125 0.7314260 1.235110
#>       PPER        NA        NA        NA
```

Let's combine the three approaches so we can compare them more easily with the true DGP parameter values:

```
practical_estimators <- rbind(data.frame(method = "DGP",
  estimator = c("Intercept", "Slope", "RSE"), value = c(practical_fixed,
  practical_proportional, practical_random), lower = NA,
  upper = NA), data.frame(method = "True lm", lm_validity_practical$estimators[1:3,
]), data.frame(method = "Criterion lm", lm_validity_practical_criterion$estimators
]), data.frame(method = "Criterion olp", olp_validity_practical_criterion$estimators
))

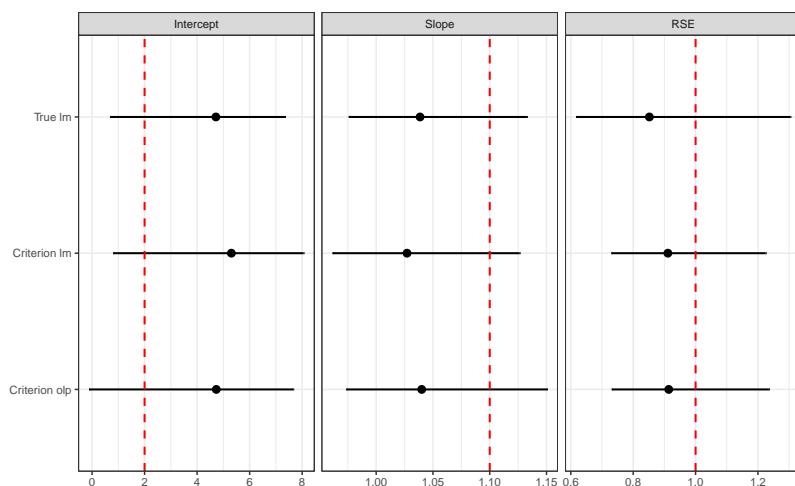
practical_estimators$method <- factor(practical_estimators$method,
  levels = rev(c("DGP", "True lm", "Criterion lm", "Criterion olp")))
```

```
practical_estimators$estimator <- factor(practical_estimators$estimator,
  levels = c("Intercept", "Slope", "RSE"))

print(practical_estimators, row.names = FALSE)
#>      method estimator    value     lower     upper
#>      DGP Intercept 2.0000000      NA      NA
#>      DGP      Slope 1.1000000      NA      NA
#>      DGP       RSE 1.0000000      NA      NA
#> True lm Intercept 4.7157542 0.6929173 7.345273
#> True lm      Slope 1.0386315 0.9758422 1.132740
#> True lm       RSE 0.8518886 0.6168049 1.304054
#> Criterion lm Intercept 5.3054383 0.8089323 8.054704
#> Criterion lm      Slope 1.0271620 0.9616332 1.126357
#> Criterion lm       RSE 0.9111659 0.7303417 1.224814
#> Criterion olp Intercept 4.7308268 -0.1090732 7.659421
#> Criterion olp      Slope 1.0400989 0.9737440 1.150441
#> Criterion olp       RSE 0.9140125 0.7314260 1.235110
```

Or plot for even more easier comparison (red vertical dashed line represent the true DGP parameter values):

```
ggplot(data = filter(practical_estimators, method != "DGP"),
  aes(y = method, x = value)) + theme_bw(8) + geom_errorbarh(aes(xmax = upper,
  xmin = lower), color = "black", height = 0) + geom_vline(data = filter(practical_estimators,
  method == "DGP"), aes(xintercept = value), linetype = "dashed",
  color = "red") + geom_point() + xlab("") + ylab("") +
  facet_wrap(~estimator, scales = "free_x")
```



To understand how these methods and how their estimates behave when there is random error in the predictor variable (x-value; criterion in this case), let's create a quick simulation (see also the SIMEX procedure explained in the [What to do when we know the error?](#) section of the [Measurement Error](#) chapter). The DGP parameters of the practical measure will stay the same, but we will change the random error for the criterion score from 0 (i.e. making it essentially the true score), to double the random error of the practical score.

```
simulation_df <- expand_grid(simulation = seq(1, 500), criterion_random_error = seq(0,
  2 * practical_random, length.out = 10))

simulation_df <- simulation_df %>% pmap_dfr(function(...) {
  current <- tibble(...)

  agreement_data <- tibble(True_score = rnorm(n_subjects,
    45, 5), Criterion_score.trial1 = 0 + (True_score *
    1) + rnorm(n_subjects, 0, current$criterion_random_error),
    Criterion_score.trial2 = 0 + (True_score * 1) + rnorm(n_subjects,
    0, current$criterion_random_error), Practical_score.trial1 = practical_fixed +
    (True_score * practical_proportional) + rnorm(n_subjects,
    0, practical_random), Practical_score.trial2 = practical_fixed +
    (True_score * practical_proportional) + rnorm(n_subjects,
    0, practical_random))

  cbind(current, agreement_data)
})

head(simulation_df)
#>   simulation criterion_random_error True_score
#> 1           1                      0  41.44284
#> 2           1                      0  42.76338
#> 3           1                      0  39.33558
#> 4           1                      0  54.14657
#> 5           1                      0  42.17742
#> 6           1                      0  50.57845
#>   Criterion_score.trial1 Criterion_score.trial2
#> 1           41.44284                  41.44284
#> 2           42.76338                  42.76338
#> 3           39.33558                  39.33558
#> 4           54.14657                  54.14657
#> 5           42.17742                  42.17742
#> 6           50.57845                  50.57845
#>   Practical_score.trial1 Practical_score.trial2
#> 1           47.36111                  47.05610
#> 2           49.30239                  47.59944
#> 3           46.81175                  45.48609
```

#> 4	62.99236	62.44907
#> 5	49.52856	49.12834
#> 6	56.14485	59.70339

Now, for each simulation, we will estimate the DGP parameters (i.e. intercept, slope, and random error) using simple linear regression and OLP regression using criterion as predictor and practical as the outcome variables.

```
estimation_wrapper <- function(data) {
  lm_true <- lm_method(data = data, criterion = "Practical_score.trial1",
    practical = "True_score", SESOI_lower = NA, SESOI_upper = NA)

  lm_criterion <- lm_method(data = data, criterion = "Practical_score.trial1",
    practical = "Criterion_score.trial1", SESOI_lower = NA,
    SESOI_upper = NA)

  olp_criterion <- olp_method(data = data, criterion = "Practical_score.trial1",
    practical = "Criterion_score.trial1", SESOI_lower = NA,
    SESOI_upper = NA)

  data.frame(simulation = data$simulation[1], criterion_random_error = data$criterion_random_error[1],
    method = c("True lm", "Criterion lm", "Criterion olp"),
    Intercept = c(lm_true[1], lm_criterion[1], olp_criterion[1]),
    Slope = c(lm_true[2], lm_criterion[2], olp_criterion[2]),
    RSE = c(lm_true[3], lm_criterion[3], olp_criterion[3]))
}

simulation_results <- simulation_df %>% group_by(simulation,
  criterion_random_error) %>% do(estimation_wrapper(.))

head(simulation_results)
#> # A tibble: 6 x 6
#> # Groups:   simulation, criterion_random_error [2]
#>   simulation criterion_random_error method Intercept Slope     RSE
#>       <int>                <dbl> <chr>      <dbl> <dbl>
#> 1          1                  0   True ~     3.35  1.08
#> 2          1                  0   Crite~     3.35  1.08
#> 3          1                  0   Crite~     2.83  1.09
#> 4          1                 0.222 True ~   -0.0301 1.14
#> 5          1                 0.222 Crite~   -0.717  1.16
#> 6          1                 0.222 Crite~   -1.38   1.17
#> # ... with 1 more variable: RSE <dbl>
```

And now we can plot the results:

```

simulation_results_long <- gather(simulation_results, "key",
    "value", -(1:3))

# Join the true DGP values for plotting
simulation_results_long <- left_join(simulation_results_long,
    data.frame(key = c("Intercept", "Slope", "RSE"), DGP = c(practical_fixed,
        practical_proportional, practical_random)), by = "key")

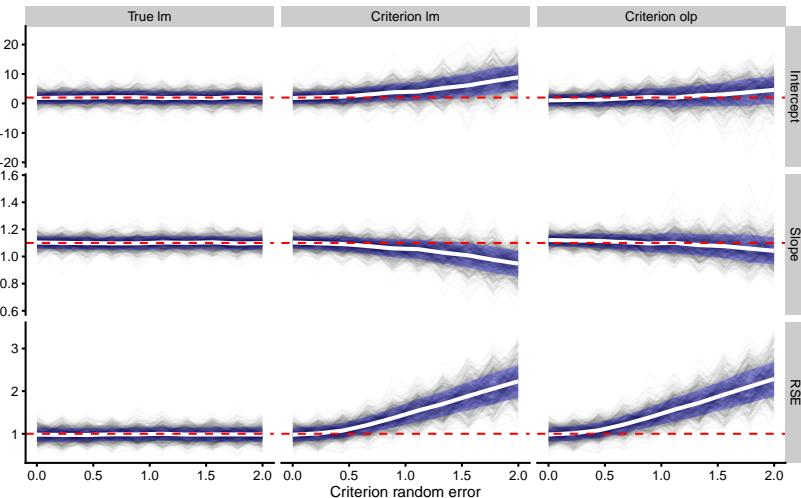
simulation_results_long$key <- factor(simulation_results_long$key,
    levels = c("Intercept", "Slope", "RSE"))

simulation_results_long$method <- factor(simulation_results_long$method,
    levels = c("True lm", "Criterion lm", "Criterion olp"))

simulation_results_long_avg <- simulation_results_long %>%
    group_by(method, criterion_random_error, key) %>% summarise(mean = mean(value),
    upper = mean + (sd(value)), lower = mean - (sd(value)))

ggplot(simulation_results_long, aes(x = criterion_random_error,
    y = value, group = simulation)) + theme_cowplot(8) +
    geom_line(alpha = 0.02) + geom_ribbon(data = simulation_results_long_avg,
    aes(y = mean, ymin = lower, ymax = upper, group = 1),
    alpha = 0.3, fill = "blue") + geom_line(data = simulation_results_long_avg,
    color = "white", aes(y = mean, group = 1), size = 1) +
    facet_grid(key ~ method, scales = "free") + geom_hline(aes(yintercept = DGP),
    linetype = "dashed", color = "red") + ylab(NULL) + xlab("Criterion random error")

```



Red dashed horizontal line on the graph indicate the true DGP parameter value,

which we want to estimate. Thin black lines (spaghetti anyone?) indicate simulation results across different levels of random error in the criterion measure. These black lines are summarized with `mean` (thick white line) \pm `SD` (blue ribbon).

As can be seen, the effect of changing random error in the criterion measure (predictor; x-value), while keeping the same random error in the practical measure (outcome) affects intercept, slope and random error estimates for criterion linear model and criterion OLP model. Intercept, slope, and `RSE` are only correctly estimated with simple regression model when there is no random error (0 on the x-axis). As the random error increases, both bias (i.e. simulations `mean`) and variance (i.e. simulations `SD`) are increased.

When it comes to OLP, due to random error in the practical measure, bias is involved in the intercept and slope estimates when there is no random error in the criterion. Intercept and slope are correctly estimated only when criterion has the same amount of random error as in the practical measure. OLP estimated `RSE` suffers the same issues as `RSE` estimated with simple linear model. Estimating Type I errors using the bootstrapped CIs would involve slower/longer simulation and will not be considered here. Although I can speculate, as can be seen from the graphs, that these will not be constants, particularly for the biased estimators.

What can we do? Well, if we know the random error involved in the criterion score (i.e. predictor) we can *adjust* slope coefficient (see *attenuation effect* and slope adjustment explained in great article by Michael Wallace (Wallace 2020, 17)) or we can perform SIMEX as explained in the [Measurement Error](#) chapter. Let's repeat the simulation, but now let's use adjusted slope and `RSE`. `RSE` is adjusted by deducting known random error: *adjusted RSE*² = *estimated RSE*² – *known random error*². The problem emerges when known random error is larger from the estimated `RSE`, since we are taking a square root of their squared difference. We thus need to make a function that deals with that:

```
adjust_RSE <- function(est_RSE, known_error) {
  ifelse(est_RSE > known_error, sqrt(est_RSE^2 - known_error^2),
        -sqrt(known_error^2 - est_RSE^2))
}
```

Let's summarize our simulation using adjusted slope and `RSE`:

```
estimation_wrapper <- function(data) {

  # Used for slope adjustment
  sd_predictor <- sd(data$Criterion_score.trial1)
  criterion_random_error <- data$criterion_random_error[1]

  slope_adj <- (sd_predictor^2 + criterion_random_error^2)/sd_predictor^2
```

```

lm_true <- lm_method(data = data, criterion = "Practical_score.trial1",
                      practical = "True_score", SESOI_lower = NA, SESOI_upper = NA)

lm_criterion <- lm_method(data = data, criterion = "Practical_score.trial1",
                           practical = "Criterion_score.trial1", SESOI_lower = NA,
                           SESOI_upper = NA)

olp_criterion <- olp_method(data = data, criterion = "Practical_score.trial1",
                             practical = "Criterion_score.trial1", SESOI_lower = NA,
                             SESOI_upper = NA)

data.frame(simulation = data$simulation[1], criterion_random_error = data$criterion_random_error,
           method = c("True lm", "Criterion lm", "Criterion olp"),
           Intercept = c(lm_true[1], lm_criterion[1], olp_criterion[1]),
           Slope = c(lm_true[2], lm_criterion[2] * slope_adj,
                     olp_criterion[2] * slope_adj), RSE = c(lm_true[3],
                     adjust_RSE(lm_criterion[3], criterion_random_error),
                     adjust_RSE(olp_criterion[3], criterion_random_error)))
}

simulation_results <- simulation_df %>% group_by(simulation,
                                                    criterion_random_error) %>% do(estimation_wrapper(.))

head(simulation_results)
#> # A tibble: 6 x 6
#> # Groups:   simulation, criterion_random_error [2]
#>   simulation criterion_random_error method Intercept Slope
#>   <int>                <dbl> <chr>      <dbl> <dbl>
#> 1          1                 0  True ~     3.35    1.08
#> 2          1                 0  Crite~     3.35    1.08
#> 3          1                 0  Crite~     2.83    1.09
#> 4          1                0.222 True ~    -0.0301  1.14
#> 5          1                0.222 Crite~    -0.717   1.16
#> 6          1                0.222 Crite~    -1.38    1.17
#> # ... with 1 more variable: RSE <dbl>

```

And we can plot the results of adjusted slope and RSE:

```

simulation_results_long <- gather(simulation_results, "key",
                                    "value", -(1:3))

# Join the true DGP values for plotting
simulation_results_long <- left_join(simulation_results_long,
                                       data.frame(key = c("Intercept", "Slope", "RSE"), DGP = c(practical_fixed,

```

```

practical_proportional, practical_random)), by = "key")

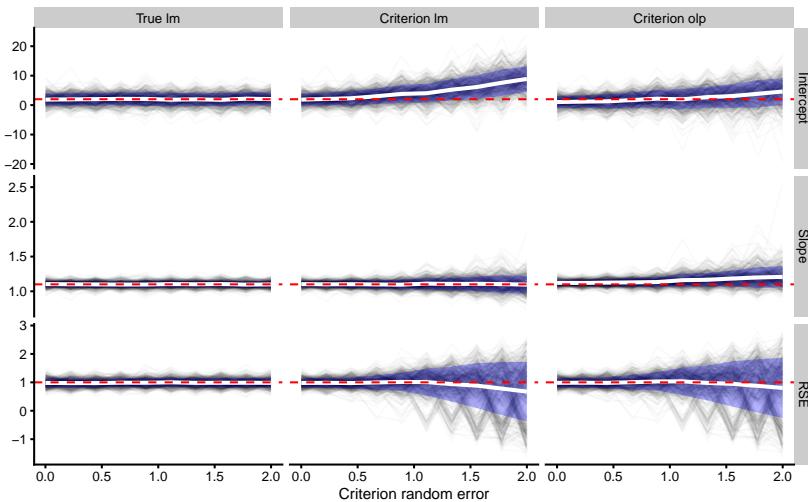
simulation_results_long$key <- factor(simulation_results_long$key,
  levels = c("Intercept", "Slope", "RSE"))

simulation_results_long$method <- factor(simulation_results_long$method,
  levels = c("True lm", "Criterion lm", "Criterion olp"))

simulation_results_long_avg <- simulation_results_long %>%
  group_by(method, criterion_random_error, key) %>% summarise(mean = mean(value),
  upper = mean + (sd(value)), lower = mean - (sd(value)))

ggplot(simulation_results_long, aes(x = criterion_random_error,
  y = value, group = simulation)) + theme_cowplot(8) +
  geom_line(alpha = 0.02) + geom_ribbon(data = simulation_results_long_avg,
  aes(y = mean, ymin = lower, ymax = upper, group = 1),
  alpha = 0.3, fill = "blue") + geom_line(data = simulation_results_long_avg,
  color = "white", aes(y = mean, group = 1), size = 1) +
  facet_grid(key ~ method, scales = "free") + geom_hline(aes(yintercept = DGP),
  linetype = "dashed", color = "red") + ylab(NULL) + xlab("Criterion random error")

```



Using the simple linear regression, we managed to adjust the bias of the slope estimate, although variance still increases with increasing criterion random error (which will probably affect the Type I error rates). Adjustment didn't help the OLP estimate slope. Adjusting RSE seems to remove the bias for both simple linear regression and OLP, but simulation variance keeps increasing with increase in criterion random error.

As stated beforehand, to estimate bootstrap CIs Type I error-rates, simulation would need to be much computation intensive, since for each simulation we would need to provide bootstrapped CIs to check their coverage (i.e. if they capture the true DGP parameter value or not). I suggest you try that as an example and leave the computer to run for few hours. Playing with these concepts (and simulations) is very helpful to understand statistics (and appreciate its complexity).

This example indicates that it is not that straight forward to re-create DGP parameters of the practical score using the criterion score, even with the known random error. But, at practitioners are we really interested in estimating DGP parameters of the measures?

15.2.3 Prediction approach

Rather than trying to re-create DGP, we might be interested in predictive performance instead. This implies estimating *predictive validity* using *calibration model* (i.e. simple linear regression used so far represent simple calibration model). Sometimes certain measuring devices produce multiple outputs, for example heat, humidity, and pressure readings as well as non-linear readings and we can use all these features as predictors to find a calibration model that provides the best predictive performance.

In our case, we are interested in predicting criterion measure from practical measure. Rather than using `lm_method` we have written, we can use *default estimator function* `bmbstats::validity_estimators`:

```
lm_criterion_validity <- bmbstats::validity_analysis(data = agreement_data,
  practical = "Practical_score.trial1", criterion = "Criterion_score.trial1",
  SESOI_lower = -2.5, SESOI_upper = 2.5, control = model_control(seed = 1667))
#> [1] "All values of t are equal to 2.5 \n Cannot calculate confidence intervals"
#> [1] "All values of t are equal to 5 \n Cannot calculate confidence intervals"

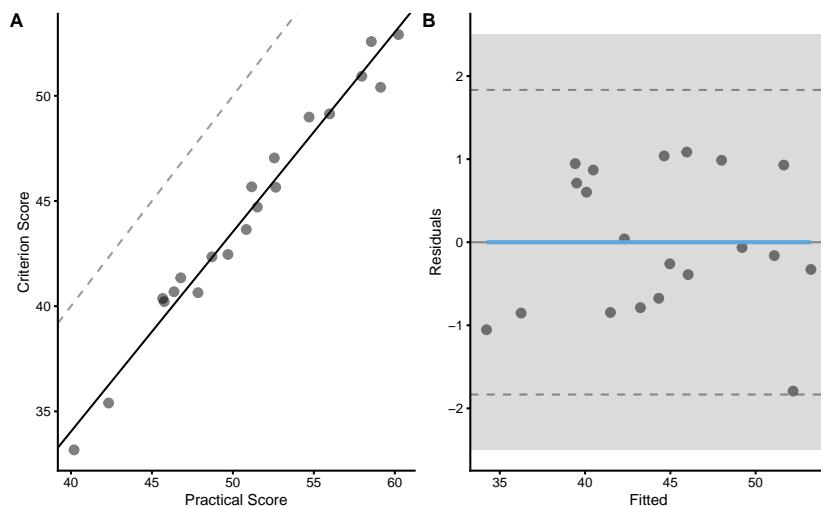
lm_criterion_validity
#> Bootstrap with 2000 resamples and 95% bca confidence intervals.
#>
#>   estimator      value      lower      upper
#>   SESOI lower -2.5000000     NA       NA
#>   SESOI upper  2.5000000     NA       NA
#>   SESOI range  5.0000000     NA       NA
#>   Intercept -3.9394072 -7.2953389  0.7734320
#>   Slope      0.9494885  0.8579138  1.0168839
#>   RSE        0.8760378  0.7261416  1.1122760
#>   Pearson's r 0.9875619  0.9763234  0.9935317
#>   R Squared   0.9752786  0.9532061  0.9871048
#>   SESOI to RSE 5.7075161  4.4952872  6.8897943
```

```
#>      PPER  0.9898416  0.9633314  0.9972420
#>      SDC   1.8335682  1.5198318  2.3280204
```

Additional estimator in this list is SDC. SDC is the *smallest detectable change* (see Smallest Detectable Change section in Measurement Error chapter), and in this case represents the smallest change in the practical measure for which we have at least 95% confidence it involves change in the criterion score. In other words, SDC represents a 95% *coverage* in the criterion for the same value of practical measure (or same values of the predictors involved, in the case when there is multiple of them). SDC is calculated using RSE and critical threshold using t-distribution to get a 95% coverage (or simply by multiplying RSE with ± 1.96).

This is better seen and understood using the residual graph on the panel B below:

```
bmbstats::plot_pair_lm(predictor = agreement_data$Practical_score.trial1,
                      outcome = agreement_data$Criterion_score.trial1, predictor_label = "Practical Score",
                      outcome_label = "Criterion Score", SESOI_lower = -2.5,
                      SESOI_upper = 2.5)
```



SDC, or *Level of Agreement* used in Bland-Altman analysis, is depicted with two horizontal dashed lines. Since these lines are within SESOI bands, this implies that practical measure has outstanding practical prediction validity (after calibration with simple linear regression, in this case to correct for fixed and proportional biases).

“But hold your horses Mladen, we haven’t tested model predictions on hold-out or unseen data!” And you are completely right. Let’s do that using `bmbstats::cv_model`:

```
lm_predictive_validity <- bmbstats::cv_model(Criterion_score.trial1 ~
  Practical_score.trial1, data = agreement_data, SESOI_lower = -2.5,
  SESOI_upper = 2.5, control = model_control(cv_folds = 5,
  cv_repeats = 10))

lm_predictive_validity
#> Training data consists of 2 predictors and 20 observations. Cross-Validation of the
#>
#> Model performance:
#>
#>     metric      training training.pooled
#>     MBE   1.882940e-14  -3.925753e-15
#>     MAE   7.209755e-01   7.106871e-01
#>     RMSE  8.310824e-01   8.191433e-01
#>     PPER  9.914445e-01   9.976365e-01
#>     SESOI to RMSE  6.016250e+00   6.103938e+00
#>     R-squared  9.752786e-01   9.759837e-01
#>     MinErr  -1.085374e+00  -1.302228e+00
#>     MaxErr   1.792261e+00   1.901850e+00
#>     MaxAbsErr 1.792261e+00   1.901850e+00
#>     testing.pooled      mean          SD        min
#>     0.02821925  0.01349363  0.52102607  -1.0035037
#>     0.80067569  0.79019212  0.21891902   0.2938133
#>     0.94237977  0.90134710  0.23117615   0.4147218
#>     0.99121033  0.91768067  0.05265932   0.6919549
#>     5.30571663  5.92631136  1.61587442   3.3199413
#>     0.96824238  0.83850772  0.56589940  -2.5312392
#>     -1.36413113 -0.87328158  0.40660854  -1.3641311
#>     2.34556533  0.99120816  0.71495843  -0.2737126
#>     2.34556533  1.33156462  0.43907222   0.7468639
#>     max
#>     1.0131942
#>     1.4283624
#>     1.5060507
#>     0.9885303
#>     12.0562748
#>     0.9953966
#>     0.4769940
#>     2.3455653
#>     2.3455653
```

If we check `MaxAbsErr` we can also see that the maximal absolute error is below

SESOI, even for the unseen data, which is outstanding. Using testing RMSE (i.e. mean across CV fold, which is equal to 0.9cm) we can calculate 95% SDC multiplying with 1.96 (or simple heuristic is just doubling the value), which gives us 1.8cm. This implies that using calibrated practical measure score (i.e. predicted criterion score), we are able to predict with 95% confidence at least change equal to 1.8cm, which is below our SESOI of 2.5cm. This concludes that calibrated practical measure has outstanding practical predictive validity of the criterion score and can be used in practice.

15.2.4 Can we adjust for the known criterion measure random error?

Let's use our simulated data to check what happens to intercept, slope, and RSE when changing the random error involved in the criterion (which is now a outcome variable). We can also adjust the estimated RSE by deducting known criterion random error. In this case, practical measure will be the predictor (x-variable) and criterion will be outcome (y-variable). Using estimated parameters with true score as outcome variable will be used as a reference.

```
estimation_wrapper <- function(data) {
  lm_true <- lm_method(data = data, practical = "Practical_score.trial1",
    criterion = "True_score", SESOI_lower = NA, SESOI_upper = NA)

  lm_criterion <- lm_method(data = data, practical = "Practical_score.trial1",
    criterion = "Criterion_score.trial1", SESOI_lower = NA,
    SESOI_upper = NA)

  olp_criterion <- olp_method(data = data, practical = "Practical_score.trial1",
    criterion = "Criterion_score.trial1", SESOI_lower = NA,
    SESOI_upper = NA)

  data.frame(simulation = data$simulation[1], criterion_random_error = data$criterion_random_error[1],
    method = c("True lm", "Criterion lm", "Criterion olp",
    "Adjusted lm", "Adjusted olp"), Intercept = c(lm_true[1],
    lm_criterion[1], olp_criterion[1], lm_criterion[1]),
    lm_criterion[1], olp_criterion[1]), Slope = c(lm_true[2], lm_criterion[2],
    olp_criterion[2], lm_criterion[2], olp_criterion[2]),
    lm_criterion[2], olp_criterion[2]), RSE = c(lm_true[3], lm_criterion[3], olp_criterion[3],
    lm_criterion[3], olp_criterion[3]), adjust_RSE(lm_criterion[3], data$criterion_random_error[1]),
    adjust_RSE(olp_criterion[3], data$criterion_random_error[1]))
}

simulation_results <- simulation_df %>% group_by(simulation,
criterion_random_error) %>% do(estimation_wrapper(.))
```

```
head(simulation_results)
#> # A tibble: 6 x 6
#> # Groups:   simulation, criterion_random_error [2]
#>   simulation criterion_random_error method Intercept Slope
#>   <int> <dbl> <chr> <dbl> <dbl>
#> 1 1 0 True ~ -2.10 0.909
#> 2 1 0 Criter~ -2.10 0.909
#> 3 1 0 Criter~ -2.60 0.920
#> 4 1 0 Adjus~ -2.10 0.909
#> 5 1 0 Adjus~ -2.60 0.920
#> 6 1 0.222 True ~ 1.24 0.852
#> # ... with 1 more variable: RSE <dbl>
```

And plot the results again:

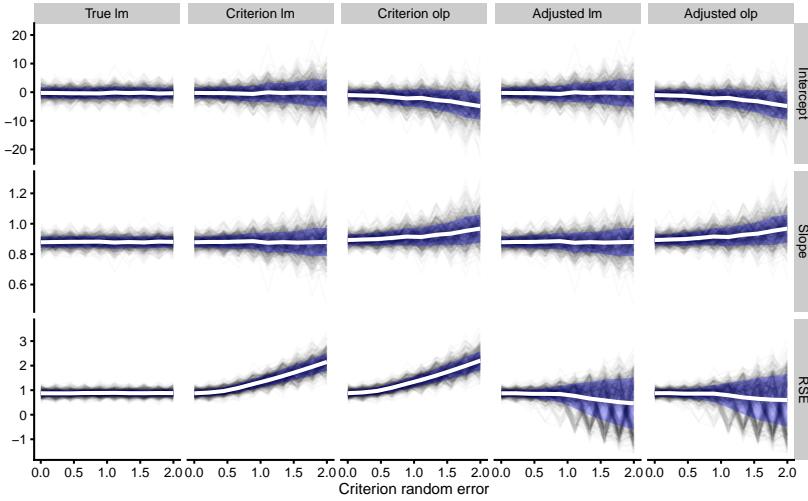
```
simulation_results_long <- gather(simulation_results, "key",
  "value", -(1:3))

simulation_results_long$key <- factor(simulation_results_long$key,
  levels = c("Intercept", "Slope", "RSE"))

simulation_results_long$method <- factor(simulation_results_long$method,
  levels = c("True lm", "Criterion lm", "Criterion olp",
  "Adjusted lm", "Adjusted olp"))

simulation_results_long_avg <- simulation_results_long %>%
  group_by(method, criterion_random_error, key) %>% summarise(mean = mean(value),
  upper = mean + (sd(value)), lower = mean - (sd(value)))

ggplot(simulation_results_long, aes(x = criterion_random_error,
  y = value, group = simulation)) + theme_cowplot(8) +
  geom_line(alpha = 0.02) + geom_ribbon(data = simulation_results_long_avg,
  aes(y = mean, ymin = lower, ymax = upper, group = 1),
  alpha = 0.3, fill = "blue") + geom_line(data = simulation_results_long_avg,
  color = "white", aes(y = mean, group = 1), size = 1) +
  facet_grid(key ~ method, scales = "free") + ylab(NULL) +
  xlab("Criterion random error")
```



As can be seen from the figure, RSE for the simple linear regression model (where criterion is the outcome and practical is the predictor) climbs up as criterion random error increases. When using adjusted RSE, up until the point where criterion has lower or equal random error to a practical score, we are able to correctly adjust RSE to give us estimation of the model fit using the true score (which is unknown of course, but we have used known criterion random error).

As can be seen from the figure as well, OLP method provided biased estimates for slope and intercept.

15.2.5 Estimating SESOI for the practical score

So far we have used SESOI of $\pm 2.5\text{cm}$ for both true score and criterion, and it was used to estimate predictive validity of the practical measure (i.e. does residuals fit within SESOI band; i.e. PPER). But we can use outcome variable SESOI to estimate SESOI in the predictor. This way we can estimate the SESOI in the predictor that yields SESOI in the criterion. In the case of simple linear regression, this is done by using estimated slope, by simply dividing outcome (i.e. criterion measure) SESOI with estimated slope:

$$\begin{aligned} SESOI_{outcome} &= \beta_1 \times predictor \\ SESOI_{predictor} &= \frac{SESOI_{outcome}}{\beta_1} \end{aligned} \quad (15.1)$$

In our validity model where criterion measure is the outcome and practical measure is the predictor, estimated slope is equal to 0.95, thus practical measure SESOI would be $\pm 2.63\text{cm}$.

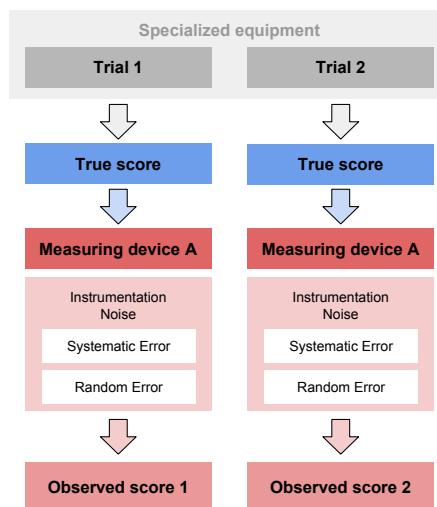
This could also be implemented inside the estimator function to provide for bootstrapped confidence intervals.

15.3 Reliability

For a measure to be valid it also needs to be reliable, but for a measure to be reliable it does not necessarily need to be valid (or at least in non-calibrated way). In essence, reliability is about *reproducibility* and *repeatability*, or how much measure is in agreement with itself. From prediction modeling perspective, reliability is how well measure predicts itself. From explanatory perspective, reliability is about estimating random error within the DGP (i.e. TE or typical error).

How is reliability evaluated? We need to make a distinction between few methods. When we have compared practical measure to criterion, we assumed criterion to be gold standard without measurement error (i.e. true score). If that analysis showed non-valid performance, it automatically implied that the measurement is unreliable. In short, if practical measure cannot predict the criterion or is not in agreement with the criterion, it cannot be reliable.

Second method is concerned with the scenario when we do not have the criterion or the true score available. Theoretically, we want a measuring instrument of interest to measure the same phenomena multiple times, to check measurement *reproducibility*. This can be achieved in few ways - we might have a special device that produce perfectly repeatable phenomena that is measured multiple times with one unit. This scenario is simulated within our DGP that we have used so far: the true score stays the same across two trials, which is then estimated with criterion and practical measures.

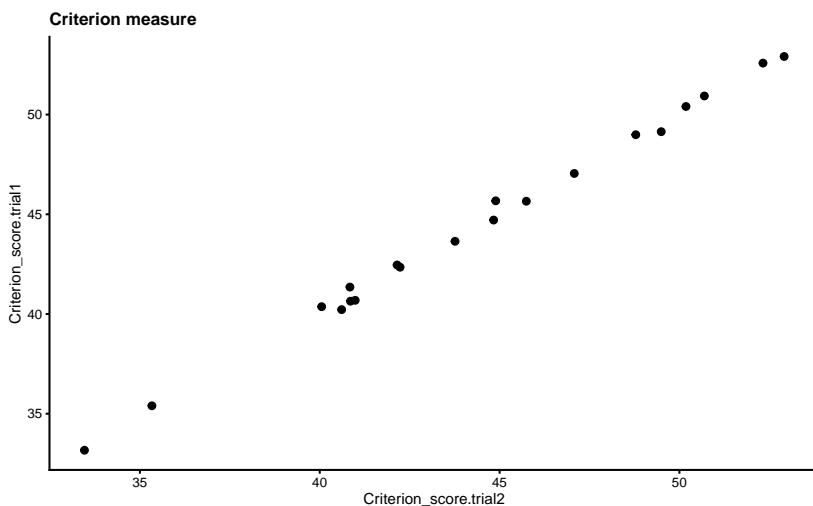


But that is easily done in simulations, and not so in real life. The other option is to measure with two or more devices of the same measuring unit. Thus, each trial is measured with two or more devices. The assumption must be made that the random error in each device is the same. Let's see how this plays out with our simulated data.

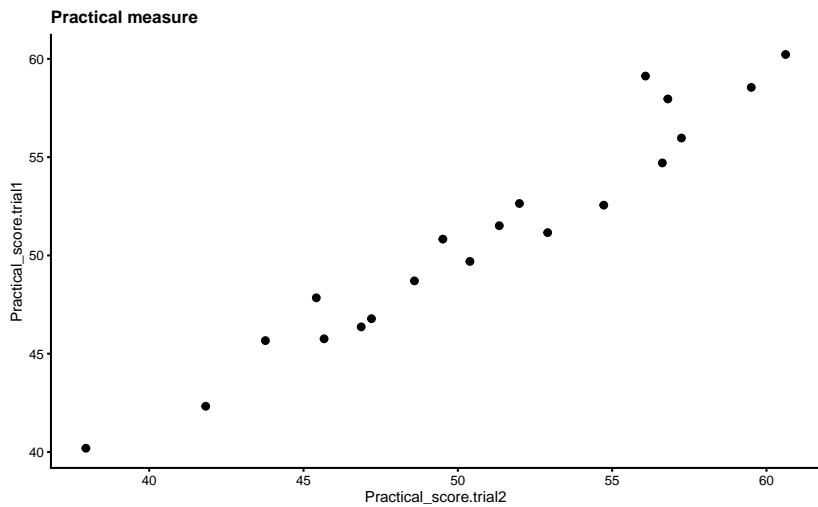
15.3.1 Reproducibility

For the purpose of this example, let's assume that true score is measured with two criterion and two practical measuring devices (or measured twice with *perfect repeatability* of the true score). Let's plot these:

```
ggplot(agreement_data, aes(x = Criterion_score.trial2, y = Criterion_score.trial1)) +
  theme_cowplot(8) + geom_point() + ggtitle("Criterion measure")
```



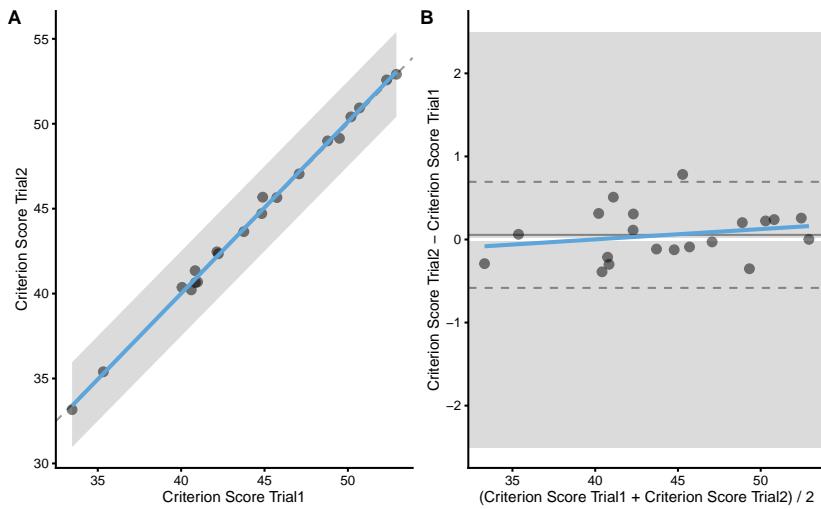
```
ggplot(agreement_data, aes(x = Practical_score.trial2, y = Practical_score.trial1)) +
  theme_cowplot(8) + geom_point() + ggtitle("Practical measure")
```



Let's check the reproducibility of the criterion measure. Same as with validity analysis, we can use few methods to estimate reproducibility: (1) method of differences (i.e. Bland-Altman), (2) simple linear regression, and (3) OLP regression. When there are more than two trials, there are few options that can be considered, and the simplest it pairwise analysis (i.e. 2-1, 3-2, 4-3 or all combinations; this technique estimates average reproducibility) or use of ANOVA or repeated-measures analysis. These will not be considered in this book.

We can use the functions that we have written already for the validity analysis. Let's start with the differences analysis by creating Bland-Altman plot:

```
bmbstats::plot_pair_BA(predictor = agreement_data$Criterion_score.trial2,
                      outcome = agreement_data$Criterion_score.trial1, predictor_label = "Criterion Score Trial2",
                      outcome_label = "Criterion Score Trial1", SESOI_lower = -2.5,
                      SESOI_upper = 2.5)
```



To provide estimators we will use functions that we have already written:

```
diff_reproducibility <- differences_method(data = agreement_data,
  criterion = "Criterion_score.trial1", practical = "Criterion_score.trial2",
  SESOI_lower = -2.5, SESOI_upper = 2.5)

diff_reproducibility
#> Mean diff      SD diff       PPER
#> 0.05560546  0.30504044  0.99999988
```

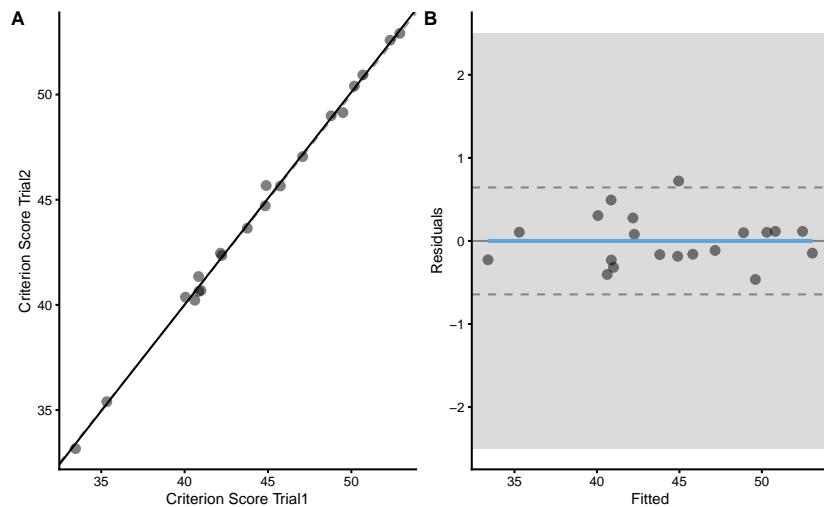
Please note the true criterion random error used in the DGP, which is equal to 0.3cm and SD diff which is not estimated with SD diff, even if they are similar: 0.31 (this is due to sampling error involved in this particular sample). As explained in [Measurement Error](#), to get the estimate of criterion score random error (i.e. TE), we need to divide SD diff with $\sqrt{2}$. This is because, random error is involved in both trials, and thus twice in their difference.

Estimated TE is equal to 0.22. If we repeat the validity analysis of the criterion score using true score as predictor, we will get the same and correct estimate:

```
differences_method(data = agreement_data, criterion = "Criterion_score.trial1",
  practical = "True_score", SESOI_lower = -2.5, SESOI_upper = 2.5)
#> Mean diff      SD diff       PPER
#> -0.07726199  0.26125825  0.99999999
```

The next approach is to use simple linear regression:

```
bmbstats::plot_pair_lm(predictor = agreement_data$Criterion_score.trial2,
  outcome = agreement_data$Criterion_score.trial1, predictor_label = "Criterion Score Trial2",
  outcome_label = "Criterion Score Trial1", SESOI_lower = -2.5,
  SESOI_upper = 2.5)
```



We can also use the function we wrote to provide validity analysis using simple linear regression:

```
lm_reproducibility <- lm_method(data = agreement_data, criterion = "Criterion_score.trial2",
  practical = "Criterion_score.trial2", SESOI_lower = -2.5,
  SESOI_upper = 2.5)

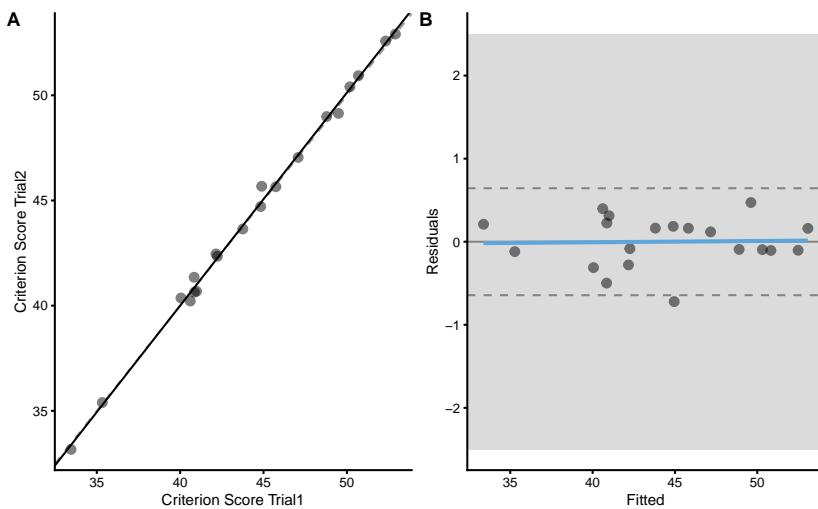
lm_reproducibility
#> Intercept      Slope       RSE        PPER
#> -0.4298105   1.0109424   0.3075601   0.9999999
```

To estimate TE, we again need to divide estimated RSE with $\sqrt{2}$.

The “problem” with both method of differences and simple linear regression, is that we need to use one variable as predictor and one as the outcome. This is avoided with the OLP approach, although parameters are estimated using one measure as predictor and other as outcome, but as explained previously, product of x- and x-variable residuals are used instead.

Let’s perform the OLP reproducibility plot and analysis using the function we have wrote already:

```
bmbstats::plot_pair_OLP(predictor = agreement_data$Criterion_score.trial2,
                        outcome = agreement_data$Criterion_score.trial1, predictor_label = "Criterion Score Trial1",
                        outcome_label = "Criterion Score Trial2", SESOI_lower = -2.5,
                        SESOI_upper = 2.5)
```



```
olp_reproducibility <- olp_method(data = agreement_data,
                                    criterion = "Criterion_score.trial1", practical = "Criterion_score.trial2",
                                    SESOI_lower = -2.5, SESOI_upper = 2.5)

olp_reproducibility
#> Intercept      Slope       RSE      PPER
#> -0.4982932   1.0124861  0.3076774  0.9999999
```

In `bmbstats`, reliability is estimated using `bmbstats::reliability_analysis` that uses OLP as the default method implemented in `bmbstats::reliability_estimators` function. Rather than using parameters `criterion` and `practical`, `bmbstats::reliability_analysis` and `bmbstats::reliability_estimators` use `trial1` and `trial2`:

```
bmbstats::reliability_estimators(data = agreement_data, trial1 = "Criterion_score.trial1",
                                  trial2 = "Criterion_score.trial2", SESOI_lower = -2.5,
                                  SESOI_upper = 2.5)
#> SESOI lower  SESOI upper  SESOI range      Intercept
#> -2.5000000   2.5000000   5.0000000  -0.4982932
#>      Slope      RSE Pearson's r      R Squared
#>  1.0124861   0.3076774   0.9984753   0.9969529
#> SESOI to RSE      PPER        TE      SDC
#> 16.2507895   0.9999999   0.2175607   0.6439761
```

`bmbstats::reliability_estimators` provides additional estimators, including TE and SDC. To get 95% bootstrap confidence intervals, use:

```
criterion_reproducibility <- bmbstats::reliability_analysis(data = agreement_data,
  trial1 = "Criterion_score.trial1", trial2 = "Criterion_score.trial2",
  SESOI_lower = -2.5, SESOI_upper = 2.5, control = model_control(seed = 1667))
#> [1] "All values of t are equal to 2.5 \n Cannot calculate confidence intervals"
#> [1] "All values of t are equal to 5 \n Cannot calculate confidence intervals"

criterion_reproducibility
#> Bootstrap with 2000 resamples and 95% bca confidence intervals.
#>
#>   estimator      value    lower     upper
#>   SESOI lower -2.5000000      NA      NA
#>   SESOI upper  2.5000000      NA      NA
#>   SESOI range   5.0000000      NA      NA
#>   Intercept -0.4982932 -1.3450623  0.6000301
#>   Slope     1.0124861  0.9888145  1.0317365
#>   RSE       0.3076774  0.2362124  0.4474032
#>   Pearson's r  0.9984753  0.9956386  0.9993848
#>   R Squared   0.9969529  0.9912963  0.9987701
#>   SESOI to RSE 16.2507895 11.2407775 21.2231826
#>   PPER       0.9999999  0.9999782  1.0000000
#>   TE        0.2175607  0.1670274  0.3163618
#>   SDC        0.6439761  0.4943982  0.9364256
```

15.3.1.1 Simulating effects of different levels of random error on reproducibility results

Since we already have generated simulated data, let's check how estimated TE behaves using three different analysis methods over varying degrees of random error involved in the criterion score. To represent the intercept for the differences method, I've used Mean Diff.

```
estimation_wrapper <- function(data) {
  diff_reporoducibility <- differences_method(data = data,
    criterion = "Criterion_score.trial1", practical = "Criterion_score.trial2",
    SESOI_lower = -2.5, SESOI_upper = 2.5)

  lm_reporoducibility <- lm_method(data = data, criterion = "Criterion_score.trial1",
    practical = "Criterion_score.trial2", SESOI_lower = -2.5,
    SESOI_upper = 2.5)

  olp_reporoducibility <- olp_method(data = data, criterion = "Criterion_score.trial1",
    practical = "Criterion_score.trial2", SESOI_lower = -2.5,
```

```

SESOI_upper = 2.5)

data.frame(simulation = data$simulation[1], criterion_random_error = data$criterion_random_error,
           method = c("diff", "lm", "olp"), Intercept = c(diff_reporoducibility[1],
               lm_reporoducibility[1], olp_reporoducibility[1]),
           Slope = c(NA, lm_reporoducibility[2], olp_reporoducibility[2]),
           TE = c(diff_reporoducibility[2], lm_reporoducibility[3],
               olp_reporoducibility[3])/sqrt(2))
}

simulation_results <- simulation_df %>% group_by(simulation,
criterion_random_error) %>% do(estimation_wrapper(.))

head(simulation_results)
#> # A tibble: 6 x 6
#> # Groups:   simulation, criterion_random_error [2]
#>   simulation criterion_random_error method Intercept Slope
#>   <int>                <dbl> <chr>      <dbl> <dbl>
#> 1          1            0.        diff     0.       NA
#> 2          1            0.        lm      -1.27e-14 1.
#> 3          1            0.        olp      0.        1
#> 4          1            0.222    diff     -8.06e- 2 NA
#> 5          1            0.222    lm      1.12e+ 0  0.974
#> 6          1            0.222    olp      1.06e+ 0  0.975
#> # ... with 1 more variable: TE <dbl>

```

And we can finally plot the results:

```

simulation_results_long <- gather(simulation_results, "key",
                                    "value", -(1:3))

# Join the true DGP values for plotting
simulation_results_long <- left_join(simulation_results_long,
                                       data.frame(key = c("Intercept", "Slope", "TE"),
                                                   DGP = c(0,
                                                       1, NA)), by = "key")

simulation_results_long$key <- factor(simulation_results_long$key,
                                         levels = c("Intercept", "Slope", "TE"))

simulation_results_long$method <- factor(simulation_results_long$method,
                                         levels = c("diff", "lm", "olp"))

simulation_results_long_avg <- simulation_results_long %>%
  group_by(method, criterion_random_error, key) %>% summarise(mean = mean(value)),

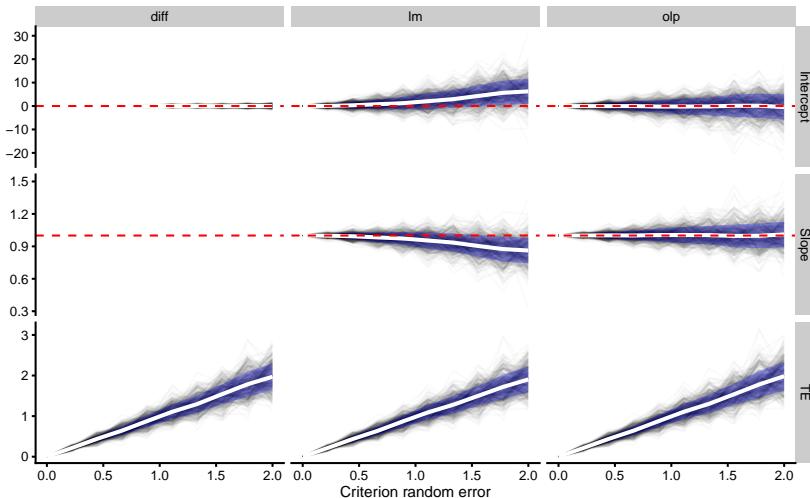
```

```

upper = mean + (sd(value)), lower = mean - (sd(value)))

ggplot(simulation_results_long, aes(x = criterion_random_error,
y = value, group = simulation)) + theme_cowplot(8) +
geom_line(alpha = 0.02) + geom_ribbon(data = simulation_results_long_avg,
aes(y = mean, ymin = lower, ymax = upper, group = 1),
alpha = 0.3, fill = "blue") + geom_line(data = simulation_results_long_avg,
color = "white", aes(y = mean, group = 1), size = 1) +
facet_grid(key ~ method, scales = "free") + geom_hline(aes(yintercept = DGP),
linetype = "dashed", color = "red") + ylab(NULL) + xlab("Criterion random error")

```

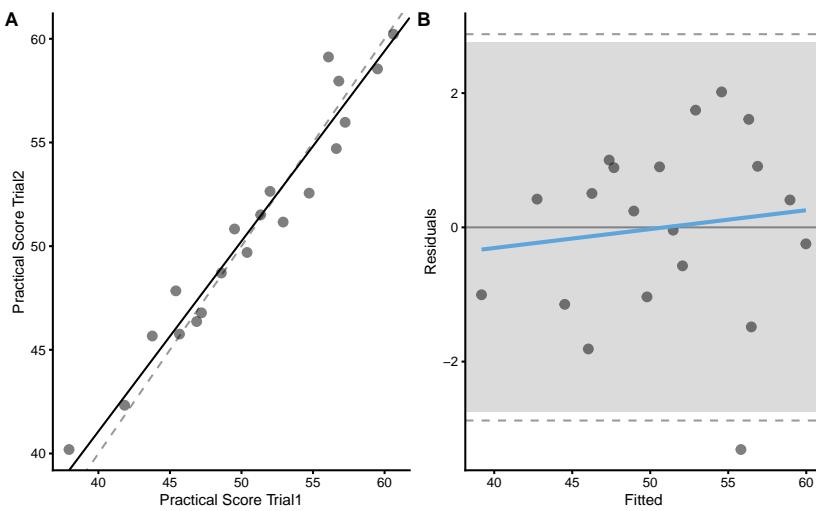


Although all three methods are good in estimating TE, only OLP should be used when estimating reproducibility fixed and proportional biases (i.e. intercept and slope; which are simulated to be 0 and 1 respectively, since both criterion measure trials have same fixed and proportional biases against the true score). Again, this recommendation is given under the assumption that the random error involved in two units/trials is the same.

15.3.1.2 Reproducibility of the practical measure

Let's quickly perform the reproducibility of the practical measure as well. SESOI for the practical measure will be equal to SESOI of the true and practical (i.e. 2.5cm) multiplied by practical proportional bias ($1.1 \times 2.5\text{cm}$). We can either use this known and true DGP relationship, or use estimated practical measure SESOI from the validity analysis.

```
bmbstats::plot_pair_OLP(predictor = agreement_data$Practical_score.trial2,
  outcome = agreement_data$Practical_score.trial1, predictor_label = "Practical Score Trial1",
  outcome_label = "Practical Score Trial2", SESOI_lower = -2.5 *
  practical_proportional, SESOI_upper = 2.5 * practical_proportional)
```



```
practical_reproducibility <- bmbstats::reliability_analysis(data = agreement_data,
  trial1 = "Practical_score.trial1", trial2 = "Practical_score.trial2",
  SESOI_lower = -2.5 * practical_proportional, SESOI_upper = 2.5 *
  practical_proportional, control = model_control(seed = 1667))
#> [1] "All values of t are equal to 2.75 \n Cannot calculate confidence intervals"
#> [1] "All values of t are equal to 5.5 \n Cannot calculate confidence intervals"

practical_reproducibility
#> Bootstrap with 2000 resamples and 95% bca confidence intervals.
#>
#>      estimator      value      lower      upper
#> SESOI lower -2.7500000      NA      NA
#> SESOI upper  2.7500000      NA      NA
#> SESOI range   5.5000000      NA      NA
#> Intercept    4.3847186 -2.3423560  8.0714488
#> Slope        0.9170754  0.8422666  1.0547194
#> RSE          1.3749629  0.9923160  2.0538771
#> Pearson's r  0.9718531  0.9378131  0.9877429
#> R Squared    0.9444984  0.8794861  0.9756338
#> SESOI to RSE 4.0001078  2.6825278  5.5671995
#> PPER         0.9400042  0.8036046  0.9875717
#> TE           0.9722456  0.7016734  1.4523105
#> SDC          2.8778305  2.0769413  4.2988143
```

15.3.1.3 Uses of reproducibility analyses

There are few uses of reproducibility analyses. If single unit is used over two (or more) trial using specialized equipment, we would tend not to expect any biases involved. If there are biases involved, the preparation and utilization of the measuring devices might be erroneous, or there might be error in the specialized equipment (i.e. indication of *systematic effects*). In either way, we do not want to see fixed nor proportional biases.

If multiple units are used to measure single phenomena, existence of biases might demand that the units needs to be *calibrated*. If that is not possible, this implies that exchange of units for longitudinal monitoring is not recommended (depends on the magnitude of the biases and differences). In plain English - you need to stick to one unit for longitudinal monitoring. Assuming, of course acceptable practical reproducibility (i.e. small TE, or SDC compared to SESOI).

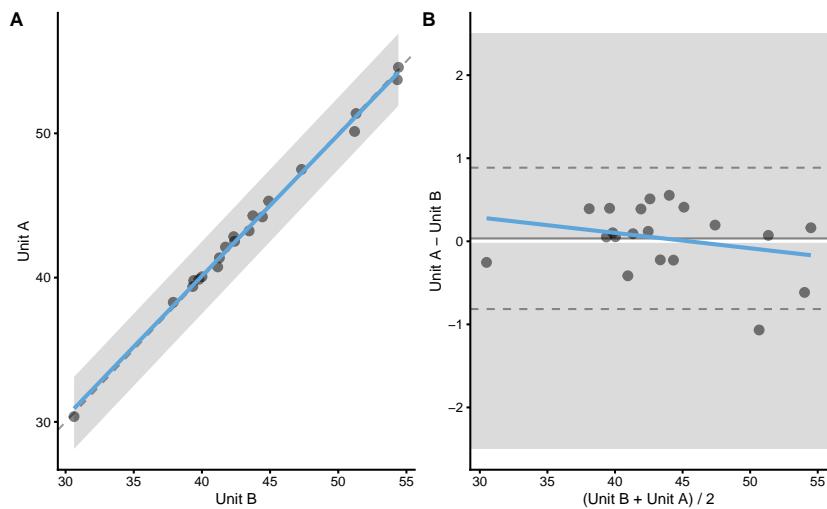
To demonstrate this, let's consider the following example. We will have three devices that measure a single true score. Two of these devices (A and B) have same biases against the true score (hence, no biases between them), while one device (device C) has different biases against the true score.

```
reproducibility_data <- tibble(Athlete = paste("Athlete",
  str_pad(string = seq(1, n_subjects), width = 2, pad = "0")),
  True_score = rnorm(n_subjects, 45, 5), Criterion_score_A = 0 +
    (True_score * 1) + rnorm(n_subjects, 0, criterion_random),
  Criterion_score_B = 0 + (True_score * 1) + rnorm(n_subjects,
    0, criterion_random), Criterion_score_C = 2.5 + (True_score *
    1.5) + rnorm(n_subjects, 0, criterion_random))

head(reproducibility_data)
#> # A tibble: 6 x 5
#>   Athlete True_score Criterion_score~ Criterion_score~
#>   <chr>     <dbl>          <dbl>          <dbl>
#> 1 Athlet~    41.4           41.4           41.3
#> 2 Athlet~    42.8           42.8           42.3
#> 3 Athlet~    39.3           39.8           39.4
#> 4 Athlet~    54.1           54.6           54.4
#> 5 Athlet~    42.2           42.5           42.4
#> 6 Athlet~    50.6           50.1           51.2
#> # ... with 1 more variable: Criterion_score_C <dbl>
```

Rather than using simple linear regression or OLP for plotting the residuals (i.e. calibrating the predictor), we will use the *raw* values using Bland-Altman plot. Here is the plot and the analysis using method of differences:

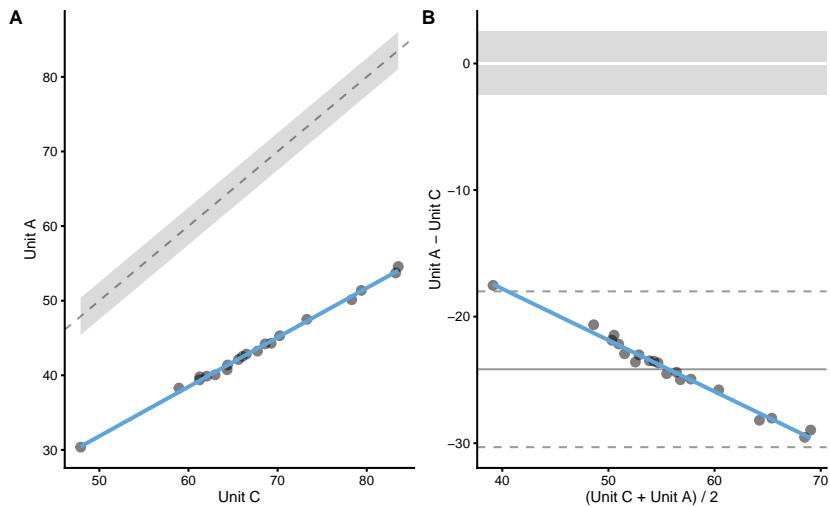
```
bmbstats::plot_pair_BA(predictor = reproducibility_data$Criterion_score_B,
                      outcome = reproducibility_data$Criterion_score_A, predictor_label = "Unit B",
                      outcome_label = "Unit A", SESOI_lower = -2.5, SESOI_upper = 2.5)
```



```
differences_method(data = reproducibility_data, criterion = "Criterion_score_A",
                     practical = "Criterion_score_B", SESOI_lower = -2.5,
                     SESOI_upper = 2.5)
#> Mean diff      SD diff       PPER
#> 0.03513462  0.40651806  0.99999335
```

As can be seen from the analysis, PPER for these two devices is excellent (and this is *without* calibration), implying that these can be used inter-changeably for longitudinal monitoring. Let's see if that is the case for unit C (comparing it with unit A):

```
bmbstats::plot_pair_BA(predictor = reproducibility_data$Criterion_score_C,
                      outcome = reproducibility_data$Criterion_score_A, predictor_label = "Unit C",
                      outcome_label = "Unit A", SESOI_lower = -2.5, SESOI_upper = 2.5)
```



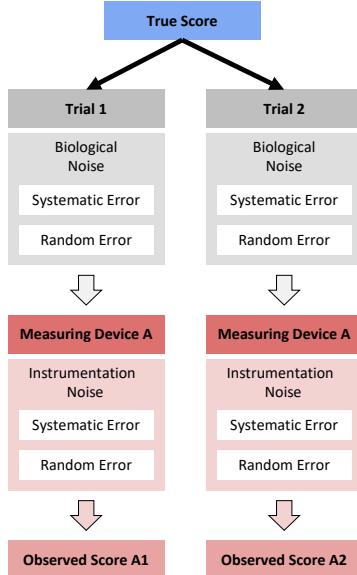
```
differences_method(data = reproducibility_data, criterion = "Criterion_score_A",
  practical = "Criterion_score_C", SESOI_lower = -2.5,
  SESOI_upper = 2.5)
#>      Mean diff        SD diff          PPER
#> -2.415796e+01  2.939988e+00  2.664952e-07
```

Unless we are able to calibrate the unit C, we cannot use it interchangeably with units A and B. But unit C is more than good to be used for measurement - we just need to stick to that unit across time.

15.4 Repeatability

So far, we have assumed no change in the true score across trials, thus no effect of the *biological noise*. In the next data set we are going to measure vertical jump height on two days spread by 7 days for N=20 athletes. We will assume that there is no change in the true score (i.e. no systematic change/effect), but there will be some *biological noise* that will affect the *manifestable performance* (see [Extending the Classical Test Theory](#) section). This unavoidably presents the question “What is the true score?”. Well, in this case we can consider it some individual *stable* level of performance, that varies from day to day due to various factors, such as sleep, nutrition, motivation, circadian rhythms and so forth. Please refer to Borsboom’s text for more in depth treatment of the issues with the *true score* concept (Borsboom 2009).

In this DGP, we will assume that this *stochastic* biological effect (i.e. random error or biological noise) is equal for all individuals (please see the [Ergodicity](#) section for more information about this assumption) and that there is not systematic effect (i.e. fatigue, learning effects and so forth).



This vertical jump is measured with our practical measuring unit, that we have shown to be reliable, and for which we have estimated TE and SDC.

```

n_subjects <- 20

practical_proportional <- 1.1
practical_random <- 1
biological_noise <- 2

set.seed(1667)

repeatability_data <- tibble(
  Athlete = paste(
    "Athlete",
    str_pad(
      string = seq(1, n_subjects),
      width = 2,
      pad = "0"
    )
  ),
  True_score.Pre = rnorm(n_subjects, 45, 5),
  True_change = rep(0, n_subjects),
  True_score.Post = True_score.Pre + True_change,

  # Add biological noise to true score
  Manifested_score.Pre = True_score.Pre + rnorm(n_subjects, 0, biological_noise),
)
  
```

```

Manifested_score.Post = True_score.Post + rnorm(n_subjects, 0, biological_noise),
Manifested_score.Change = Manifested_score.Post - Manifested_score.Pre,

# Add measurement of the manifested score
Measured_score.Pre = practical_fixed +
  (Manifested_score.Pre * practical_proportional) +
  rnorm(n_subjects, 0, practical_random),

Measured_score.Post = practical_fixed +
  (Manifested_score.Post * practical_proportional) +
  rnorm(n_subjects, 0, practical_random),

Measured_score.Change = Measured_score.Post - Measured_score.Pre
)

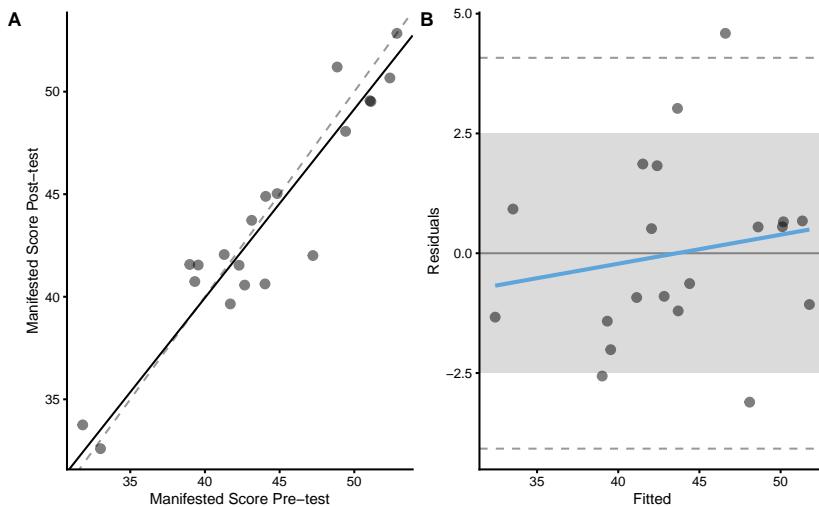
head(repeatability_data)
#> # A tibble: 6 x 10
#>   Athlete True_score.Pre True_change True_score.Post
#>   <chr>      <dbl>       <dbl>        <dbl>
#> 1 Athlet~     52.9        0          52.9
#> 2 Athlet~     42.4        0          42.4
#> 3 Athlet~     49.2        0          49.2
#> 4 Athlet~     44.8        0          44.8
#> 5 Athlet~     40.0        0          40.0
#> 6 Athlet~     42.6        0          42.6
#> # ... with 6 more variables:
#> #   Manifested_score.Pre <dbl>,
#> #   Manifested_score.Post <dbl>,
#> #   Manifested_score.Change <dbl>,
#> #   Measured_score.Pre <dbl>,
#> #   Measured_score.Post <dbl>,
#> #   Measured_score.Change <dbl>

```

As can be see in the DGP, we have three components: (1) true score, that doesn't change from Pre-test to Post-test, (2) manifested score, that doesn't have systematic effect (i.e. expected or `mean` is equal to 0) but is affected by random biological noise, and (3) measured score that is affected by both biological noise and instrumentation error (vertical jump was measured using the practical measure).

Let's estimate reliability using each of these scores. Here is the reliability of the manifested score. SESOI for both true and manifested scores is chosen to be $\pm 2.5\text{cm}$:

```
bmbstats::plot_pair_OLP(predictor = repeatability_data$Manifested_score.Pre,
  outcome = repeatability_data$Manifested_score.Post, predictor_label = "Manifested Score Pre-test",
  outcome_label = "Manifested Score Post-test", SESOI_lower = -2.5,
  SESOI_upper = 2.5)
```

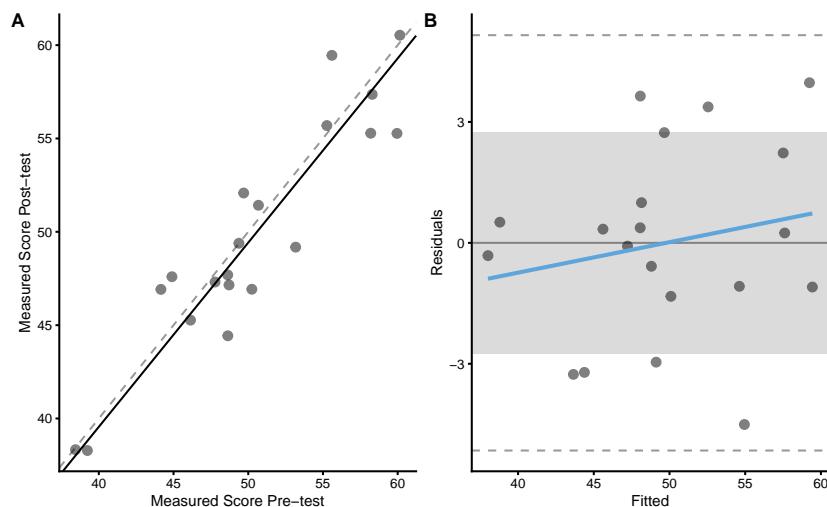


```
manifested_reliability <- bmbstats::reliability_analysis(data = repeatability_data,
  trial1 = "Manifested_score.Pre", trial2 = "Manifested_score.Post",
  SESOI_lower = -2.5, SESOI_upper = 2.5, control = model_control(seed = 1667))
#> [1] "All values of t are equal to 2.5 \n Cannot calculate confidence intervals"
#> [1] "All values of t are equal to 5 \n Cannot calculate confidence intervals"

manifested_reliability
#> Bootstrap with 2000 resamples and 95% bca confidence intervals.
#>
#>      estimator      value      lower      upper
#> SESOI lower -2.5000000      NA      NA
#> SESOI upper 2.5000000      NA      NA
#> SESOI range 5.0000000      NA      NA
#> Intercept -3.4271606 -9.5821430 2.9874926
#> Slope     1.0870906  0.9387198 1.2275893
#> RSE       2.1181964  1.6136378 3.0424513
#> Pearson's r 0.9394992  0.8345656 0.9759104
#> R Squared  0.8826588  0.6965898 0.9524020
#> SESOI to RSE 2.3604987  1.6480073 3.1187994
#> PPER      0.7475330  0.5785637 0.8622366
#> TE        1.4977911  1.1410142 2.1513380
#> SDC       4.4334361  3.3773827 6.3679238
```

Unfortunately, we do not know this manifested score, as we do not know the true score. We can only measure manifested score. The estimated TE of the practical measure will now contain both instrumentation noise and biological noise. As explained previously, the SESOI for practical is equal to true SESOI multiplied by practical proportional bias ($1.1 \times 2.5\text{cm}$):

```
bmbstats::plot_pair_OLP(predictor = repeatability_data$Measured_score.Pre,
  outcome = repeatability_data$Measured_score.Post, predictor_label = "Measured Score Pre-test",
  outcome_label = "Measured Score Post-test", SESOI_lower = -2.5 *
  practical_proportional, SESOI_upper = 2.5 * practical_proportional)
```



```
measured_reliability <- bmbstats::reliability_analysis(data = repeatability_data,
  trial1 = "Measured_score.Pre", trial2 = "Measured_score.Post",
  SESOI_lower = -2.5 * practical_proportional, SESOI_upper = 2.5 *
  practical_proportional, control = model_control(seed = 1667))
#> [1] "All values of t are equal to 2.75 \n Cannot calculate confidence intervals"
#> [1] "All values of t are equal to 5.5 \n Cannot calculate confidence intervals"

measured_reliability
#> Bootstrap with 2000 resamples and 95% bca confidence intervals.
#>
#>   estimator      value      lower      upper
#>   SESOI lower -2.750000000      NA      NA
#>   SESOI upper  2.750000000      NA      NA
#>   SESOI range  5.500000000      NA      NA
#>   Intercept -0.07812008 -9.5613061  7.5298921
#>   Slope     1.01329856  0.8553983  1.2053227
#>   RSE       2.49420427  1.9646649  3.3307725
```

```
#> Pearson's r 0.92440336 0.8391887 0.9681646
#> R Squared 0.85452158 0.7042450 0.9373512
#> SESOI to RSE 2.20511209 1.6537377 2.8007336
#> PPER 0.71600963 0.5809328 0.8223197
#> TE 1.76366875 1.3892279 2.3552118
#> SDC 5.22042954 4.1120910 6.9713869
```

As can be seen from the analysis, now the biological noise is contained (or propagated) to RSE, SDC and other estimators of the practical measure random error. The random error (TE) inside the practical measure is equal to:

$$TE^2 = (\text{proportional bias} \times \text{biological noise})^2 + \text{practical random error}^2 \quad (15.2)$$

Estimated biological noise (using manifested analysis TE) is equal to 1.5cm (or we could use true DGP value of 2cm), simulated practical random error is equal to 1cm, and the simulated slope is equal to 1.1. Thus expected estimate TE of the practical measure is equal to 1.93cm, which is very close to the above estimated TE (1.76cm).

15.5 The difference between Reproducibility and Repeatability

What's the difference between the estimate reproducibility and repeatability of the practical measure? Well, with reproducibility we aim to estimate sole instrumentation noise without biological noise/variation. This is done by making sure that the devices are measuring exactly the same phenomena.

With repeatability, we want to estimate combination of the biological variation and instrumentation noise. Since we have used SESOI defined for reproducibility (i.e. same true score), the magnitude-based estimators were much worse with the repeatability analysis, where biological variation introduced additional stochastic component.

In [Measurement Error](#) chapter, I have explained that measurement error is the combination of instrumentation noise and biological variation (i.e. repeatability analysis is used to estimate it). I have also explained the use of SDC (and TE) to interpret individual changes (see [Interpreting individual changes using SESOI and SDC](#) in the aforementioned chapter). The question is then which SDC should be used? The one estimated from reproducibility (or validity analysis), or the one estimated from repeatability (involving biological noise)?

The answer depends on what question we are trying to answer. If we use SDC from reproducibility analysis (i.e. pure instrumentation noise) to interpret individual

change, we will estimate the uncertainty around manifested performance change. On the other hand, if we use SDC from repeatability analysis (i.e. combined biological variation and instrumentation noise), we will estimate uncertainty around true performance change.

To demonstrate and plot these differences, let's use `bmbstats::observations_MET` that we have introduced in [Measurement error issues](#) section of the [Descriptive tasks using bmbstats](#) chapter. But before we demonstrate these differences, we need to create few more column inside our simulated DGP - we need to have true score as would be measured by the practical measure (without the practical measure random noise), and manifested score as would be measured by the practical measure (without the practical measure random noise). This needs to be done to deal with practical measure proportional bias.

```
repeatability_data <- repeatability_data %>%
  mutate(
    True_score_measured.Pre = practical_fixed + (True_score.Pre * practical_proportion),
    True_score_measured.Post = practical_fixed + (True_score.Post * practical_proportion),
    True_score_measured.Change = True_score_measured.Post - True_score_measured.Pre,

    Manifested_score_measured.Pre = practical_fixed + (Manifested_score.Pre * practical_proportion),
    Manifested_score_measured.Post = practical_fixed + (Manifested_score.Post * practical_proportion),
    Manifested_score_measured.Change = Manifested_score_measured.Post - Manifested_score_measured.Pre
  )

head(repeatability_data)
#> # A tibble: 6 x 16
#>   Athlete True_score.Pre True_change True_score.Post
#>   <chr>      <dbl>        <dbl>        <dbl>
#> 1 Athlet~     52.9         0          52.9
#> 2 Athlet~     42.4         0          42.4
#> 3 Athlet~     49.2         0          49.2
#> 4 Athlet~     44.8         0          44.8
#> 5 Athlet~     40.0         0          40.0
#> 6 Athlet~     42.6         0          42.6
#> # ... with 12 more variables:
#> #   Manifested_score.Pre <dbl>,
#> #   Manifested_score.Post <dbl>,
#> #   Manifested_score.Change <dbl>,
#> #   Measured_score.Pre <dbl>,
#> #   Measured_score.Post <dbl>,
#> #   Measured_score.Change <dbl>,
#> #   True_score_measured.Pre <dbl>,
#> #   True_score_measured.Post <dbl>,
#> #   True_score_measured.Change <dbl>,
#> #   Manifested_score_measured.Pre <dbl>,
```

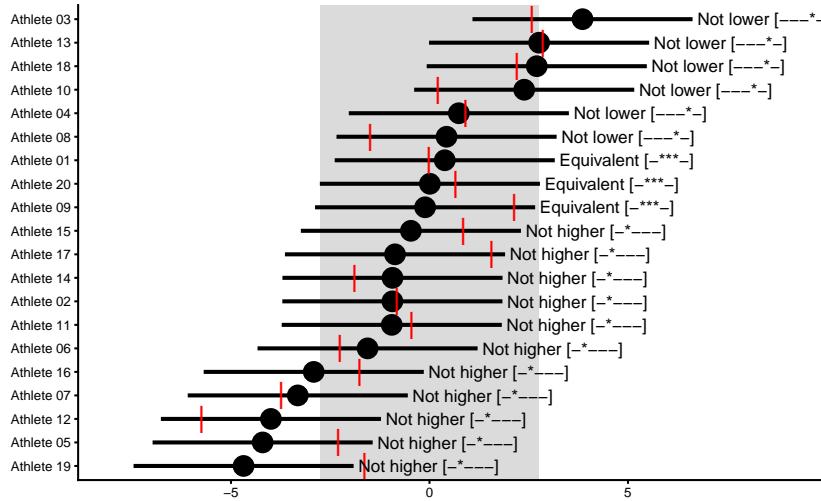
```
#> # Manifested_score_measured.Post <dbl>,
#> # Manifested_score_measured.Change <dbl>
```

Now let's plot measured change (i.e. difference between measured Post-test and Pre-test using practical measure), together with manifested change (as would be measured with practical measure with no random error). Confidence intervals around measured change represent 95% SDC that is calculated using known practical measure random error (i.e. estimated TE from reproducibility analysis - the sole instrumentation noise). CIs should capture vertical red lines (manifested change):

```
measured_change <- bmbstats::observations_MET(
  observations = repeatability_data$Measured_score.Change,
  observations_label = repeatability_data$Athlete,
  measurement_error = practical_random * sqrt(2),
  # Degrees of freedom from the reliability study. Use `Inf` for normal distribution
  df = Inf,
  SESOI_lower = -2.5 * practical_proportional,
  SESOI_upper = 2.5 * practical_proportional,

  # Will not use Bonferroni adjustment here
  alpha = 0.05,
  # No adjustment in CIs for plotting
  confidence = 0.95
)

plot(
  measured_change,
  true_observations = repeatability_data$Manifested_score_measured.Change,
  control = plot_control(points_size = 5)) +
  xlim(-8, 9)
```



Our CIs using sole instrumentation noise (I have used DGP parameter, but in real life we can use TE from reproducibility analysis) managed to capture manifested change (i.e. true change plus biological variation).

If we use TE from repeatability analysis (i.e. the one that captures both biological variation and instrumentation noise), we will be interested in capturing “true change” (without biological variation):

```

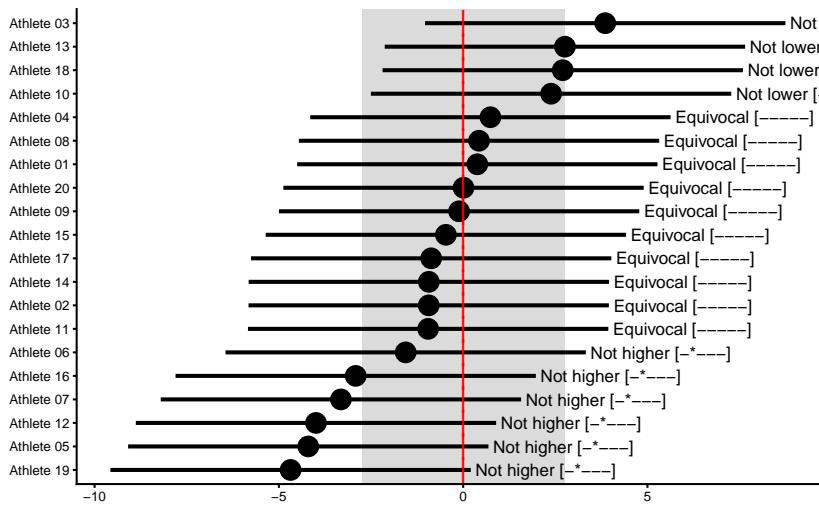
measured_change <- bmbstats::observations_MET(
  observations = repeatability_data$Measured_score.Change,
  observations_label = repeatability_data$Athlete,

  # Use TE from repeatability analysis
  measurement_error = measured_reliability$estimators$value[11] * sqrt(2),
  # Degrees of freedom from the reliability study. Use `Inf` for normal distribution
  df = Inf,
  SESOI_lower = -2.5 * practical_proportional,
  SESOI_upper = 2.5 * practical_proportional,

  # Will not use Bonferroni adjustment here
  alpha = 0.05,
  # No adjustment in CIs for plotting
  confidence = 0.95
)

plot(
  measured_change,
  true_observations = repeatability_data$True_score_measured.Change,
  control = plot_control(points_size = 5))

```



Since there is no true change in this DGP, all vertical red lines are aligned at 0. Our CIs in this case represent 95% SDC estimated using both biological variation and instrumentation error.

It is important to remember when analyzing individual change how TE or SDC are estimated - does it represent sole instrumentation error (i.e. estimated in validity or reproducibility study) or it represents measurement error combined of biological variation and instrumentation error (i.e. estimated in repeatability study). Ideally, we would need to know instrumentation noise and biological noise. If we know instrumentation noise (from validity or reproducibility study), we can *extract* biological variation (as it would be measured with the practical measure in this case).

To get biological noise, as it would be measured with the practical measure, let's use the generated columns:

```
manifested_measured_reliability <- bmbstats::reliability_analysis(data = repeatability_data,
  trial1 = "Manifested_score_measured.Pre", trial2 = "Manifested_score_measured.Post",
  SESOI_lower = -2.5 * practical_proportional, SESOI_upper = 2.5 *
  practical_proportional, control = model_control(seed = 1667))
#> [1] "All values of t are equal to 2.75 \n Cannot calculate confidence intervals"
#> [1] "All values of t are equal to 5.5 \n Cannot calculate confidence intervals"

manifested_measured_reliability
#> Bootstrap with 2000 resamples and 95% bca confidence intervals.
#>
#>   estimator      value     lower      upper
#>   SESOI lower -2.7500000      NA       NA
#>   SESOI upper  2.7500000      NA       NA
```

```
#> SESOI range 5.5000000 NA NA
#> Intercept -3.9440578 -11.0023147 3.4883182
#> Slope 1.0870906 0.9387198 1.2275893
#> RSE 2.3300161 1.7750016 3.3466965
#> Pearson's r 0.9394992 0.8345656 0.9759104
#> R Squared 0.8826588 0.6965898 0.9524020
#> SESOI to RSE 2.3604987 1.6480073 3.1187994
#> PPER 0.7475330 0.5785637 0.8622366
#> TE 1.6475702 1.2551157 2.3664718
#> SDC 4.8767797 3.7151210 7.0047162
```

Measured manifested TE represents biological noise/variation that would be measured with the practical measure without random error. It is equal to 1.65cm. But we do not know this in the real life and we want to estimate it from the TE from repeatability study and know instrumentation noise.

The estimated TE of the practical measure in the repeatability study is equal to 1.76cm. This TE contains both biological noise and instrumentation noise. Since we know the instrumentation noise (either from reproducibility or validity study; or from DGP in our case), we can estimate biological error (as would be estimated by practical measure without random error):

$$\begin{aligned} TE^2 &= Biological^2 + Instrument^2 \\ Biological^2 &= TE^2 - Instrument^2 \\ Biological &= \sqrt{TE^2 - Instrument^2} \end{aligned} \tag{15.3}$$

We thus get the estimate of biological variation to be equal to 1.45cm, which is very close to the above estimate biological variation of 1.65cm. Please note that the difference is due to me using simulation true practical measure random error, rather than estimated in this sample.

We will continue this topic in the next chapter where we will discuss Randomized Controlled Trials and the Control group will serve as a simple repeatability study that will inform us about uncertainty around true treatment effect.

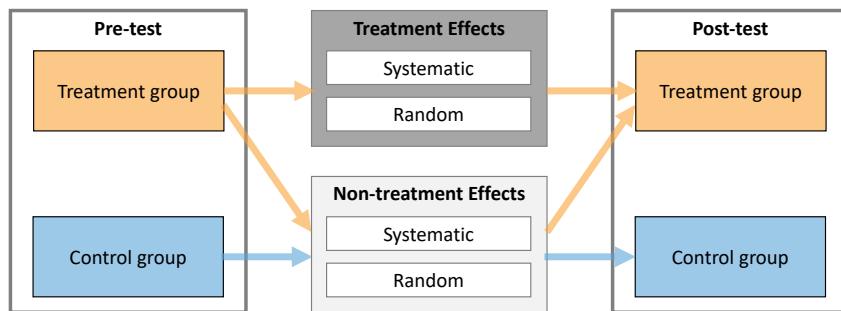
Chapter 16

RCT analysis and prediction in `bmbstats`

In this chapter I will demonstrate how to analyze simple randomized controlled trials (RCTs) from both explanatory and predictive perspectives using `bmbstats` package and functions. As a refresher, please consider re-reading [Causal inference](#) chapter.

16.1 Data Generating Process behind RCT

The following image is re-posted from the [Causal inference](#) chapter, outlining Treatment and Non-Treatment effects.



Let's consider the following RCT DGP. We have two group (Control and Treatment), each with N=10 athletes, measured twice (Pre-test and Post-test) on

the vertical jump height, using a measuring device with a known measurement error (i.e. only instrumentation noise; see [Validity and Reliability](#) chapter for more info) estimated through validity and reliability studies and equal to 0.5cm. Control group are doing their normal training for 4 weeks, while Treatment group is doing EMS stimulation of their calf muscles on top of their normal training.

Since this is DGP, we will assume there is no treatment effect nor non-treatment effects. Both Control and Treatment groups will experience normal biological variation in their jump height, which is equal to 1.5cm. Please refer to [Validity and Reliability](#) chapter for more info about the concepts of true score and measurement error. SESOI for the measured score will be $\pm 5\text{cm}$ (since there is no proportional bias in the measurement, this will also be SESOI for the true score - see [Validity and Reliability](#) chapter for more information).

```
require(tidyverse)
require(bmbstats)
require(cowplot)

set.seed(1667)

n_subjects <- 20

instrumentation_noise <- 0.5
biological_variation <- 1.5

# -----
# Treatment effect
treatment_systematic <- 0
treatment_random <- 0

# Non-treatment effect
non_treatment_systematic <- 0
non_treatment_random <- 0
#-----

RCT_data <- tibble(
  Athlete = paste(
    "Athlete",
    str_pad(
      string = seq(1, n_subjects),
      width = 2,
      pad = "0"
    )
  ),
  )
```

```

Group = rep(c("Treatment", "Control"), length.out = n_subjects),

# True score
True_score.Pre = rnorm(n_subjects, 45, 5),

# Treatment effect
Treatment_effect = rnorm(n = n_subjects, mean = treatment_systematic, sd = treatment_random),
Non_treatment_effect = rnorm(n = n_subjects, mean = non_treatment_systematic, sd = non_treatment_random)

# Calculate the change in true score
True_score.Change = if_else(
  Group == "Treatment",
  # Treatment group is a sum of treatment and non-treatment effects
  Treatment_effect + Non_treatment_effect,
  # While control group only get non-treatment effects
  Non_treatment_effect
),
True_score.Post = True_score.Pre + True_score.Change,

# Manifested score
Manifested_score.Pre = True_score.Pre + rnorm(n_subjects, 0, biological_variation),
Manifested_score.Post = True_score.Post + rnorm(n_subjects, 0, biological_variation),
Manifested_score.Change = Manifested_score.Post - Manifested_score.Pre,

# Measured score
Measured_score.Pre = Manifested_score.Pre + rnorm(n_subjects, 0, instrumentation_noise),
Measured_score.Post = Manifested_score.Post + rnorm(n_subjects, 0, instrumentation_noise),
Measured_score.Change = Measured_score.Post - Measured_score.Pre
)

# Make sure that the Group column is a factor
# This will not create issues with PDP+ICE and ICE plotting later
RCT_data$Group <- factor(RCT_data$Group)

head(RCT_data)
#> # A tibble: 6 x 13
#>   Athlete Group True_score.Pre Treatment_effect
#>   <chr>   <fct>     <dbl>          <dbl>
#> 1 Athlet~ Tre~      52.9           0
#> 2 Athlet~ Cont~     42.4           0
#> 3 Athlet~ Tre~      49.2           0
#> 4 Athlet~ Cont~     44.8           0
#> 5 Athlet~ Tre~      40.0           0
#> 6 Athlet~ Cont~     42.6           0
#> # ... with 9 more variables:

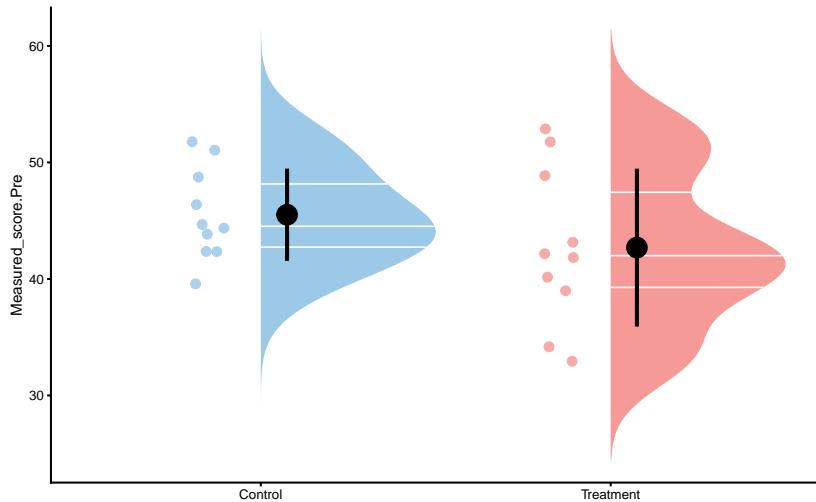
```

```
#> # Non_treatment_effect <dbl>,
#> # True_score.Change <dbl>, True_score.Post <dbl>,
#> # Manifested_score.Pre <dbl>,
#> # Manifested_score.Post <dbl>,
#> # Manifested_score.Change <dbl>,
#> # Measured_score.Pre <dbl>,
#> # Measured_score.Post <dbl>,
#> # Measured_score.Change <dbl>
```

Since we have generated this data, we know that there are no treatment nor non-treatment effects (neither systematic, nor random or variable effects). But let's plot the data.

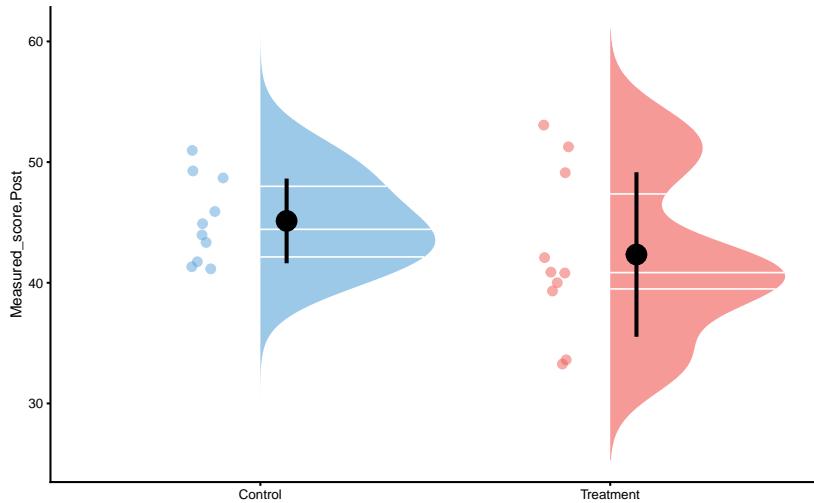
First let's plot the Pre-test measured scores (vertical jump) distribution:

```
bmbstats::plot_raincloud(RCT_data, value = "Measured_score.Pre",
                           groups = "Group")
```



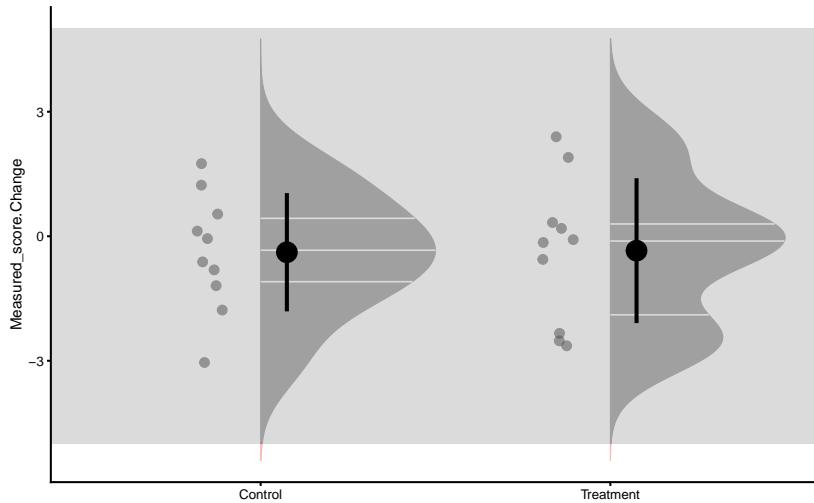
And the Post-test:

```
bmbstats::plot_raincloud(RCT_data, value = "Measured_score.Post",
                           groups = "Group")
```



And finally measured change scores:

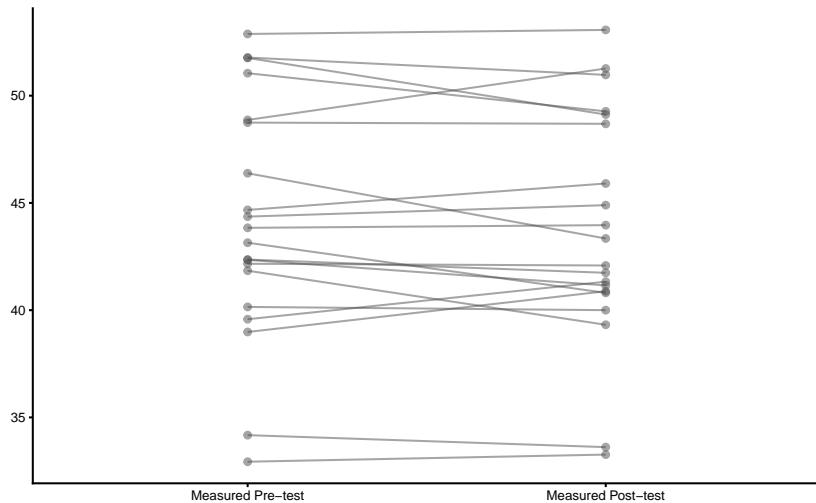
```
bmbstats::plot_raincloud_SESOI(RCT_data, value = "Measured_score.Change",
                                groups = "Group", SESOI_lower = -5, SESOI_upper = 5)
```



From these graphs we can see that there is no difference between group. We have also selected large SESOI taking into account our *a priori* knowledge about biological variation and measurement error in the vertical jump height.

Let's plot the individual change fro athletes from the Treatment group:

```
bmbstats::plot_pair_changes(group_a = RCT_data$Measured_score.Pre,
  group_b = RCT_data$Measured_score.Post, group_a_label = "Measured Pre-test",
  group_b_label = "Measured Post-test", SESOI_lower = -5,
  SESOI_upper = 5)
```



16.2 RCT analysis using `bmbstats::RCT_analysis` function

To perform RCT analysis explained in the [Causal inference](#) chapter, we will use `bmbstats::RCT_analysis` function. `bmbstats::RCT_analysis` function has three built-in estimator function: `bmbstats::RCT_estimators`, `bmbstats::RCT_estimators_simple`, and `bmbstats::RCT_estimators_lm`. `bmbstats::RCT_estimators` is more extensive and involves individual group analysis, while the `bmbstats::RCT_estimators_simple` only provides estimated treatment and non-treatment effects. `bmbstats::RCT_estimators_lm` will be explained in the next section.

```
extensive_RCT <- bmbstats::RCT_analysis(data = RCT_data,
  group = "Group", treatment_label = "Treatment", control_label = "Control",
  pre_test = "Measured_score.Pre", post_test = "Measured_score.Post",
  SESOI_lower = -5, SESOI_upper = 5, control = model_control(seed = 1667))
#> [1] "All values of t are equal to 5 \n Cannot calculate confidence intervals"
#> [1] "All values of t are equal to 10 \n Cannot calculate confidence intervals"
```

```

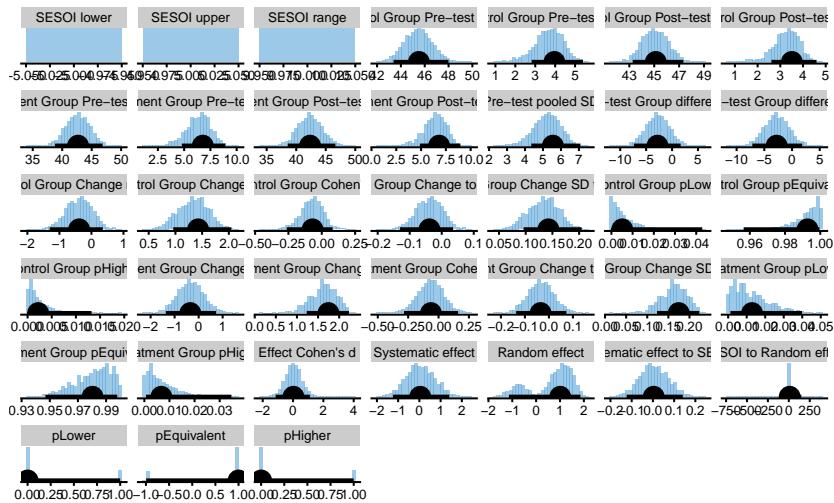
extensive_RCT
#> Bootstrap with 2000 resamples and 95% bca confidence intervals.
#>
#>           estimator      value
#> SESOI lower -5.00000000000
#> SESOI upper  5.00000000000
#> SESOI range 10.00000000000
#> Control Group Pre-test mean 45.5109521606
#> Control Group Pre-test SD 3.9597464864
#> Control Group Post-test mean 45.1257775214
#> Control Group Post-test SD 3.5092078780
#> Treatment Group Pre-test mean 42.6896373231
#> Treatment Group Pre-test SD 6.7766398945
#> Treatment Group Post-test mean 42.3426726375
#> Treatment Group Post-test SD 6.8172485979
#> Pre-test pooled SD 5.5498847059
#> Pre-test Group difference -2.8213148375
#> Post-test Group difference -2.7831048839
#> Control Group Change mean -0.3851746392
#> Control Group Change SD 1.4244343846
#> Control Group Cohen's d -0.0694022776
#> Control Group Change to SESOI -0.0385174639
#> Control Group Change SD to SESOI 0.1424434385
#> Control Group pLower 0.0050812820
#> Control Group pEquivalent 0.9927461183
#> Control Group pHIGHER 0.0021725997
#> Treatment Group Change mean -0.3469646855
#> Treatment Group Change SD 1.7452275994
#> Treatment Group Cohen's d -0.0625174583
#> Treatment Group Change to SESOI -0.0346964686
#> Treatment Group Change SD to SESOI 0.1745227599
#> Treatment Group pLower 0.0128923459
#> Treatment Group pEquivalent 0.9803630085
#> Treatment Group pHIGHER 0.0067446456
#> Effect Cohen's d 0.0268246499
#> Systematic effect 0.0382099536
#> Random effect 1.0083680171
#> Systematic effect to SESOI 0.0038209954
#> SESOI to Random effect 9.9170142553
#> pLower 0.0003713043
#> pEquivalent 0.9992167407
#> pHIGHER 0.0004119550
#> lower      upper
#> NA          NA
#> NA          NA

```

```
#>          NA          NA
#> 4.338522e+01 47.99500924
#> 2.821249e+00 5.39401519
#> 4.327530e+01 47.33738237
#> 2.612392e+00 4.59657735
#> 3.878113e+01 46.84572579
#> 4.783839e+00 8.92052081
#> 3.856045e+01 46.63546378
#> 4.880090e+00 8.96710016
#> 4.348937e+00 7.08556923
#> -7.362312e+00 1.66624873
#> -7.151974e+00 1.84048180
#> -1.301390e+00 0.35428624
#> 9.592800e-01 2.04852176
#> -2.627647e-01 0.08120665
#> -1.301390e-01 0.03542862
#> 9.592800e-02 0.20485218
#> 3.123715e-04 0.04282747
#> 9.564951e-01 0.99950198
#> 2.037526e-04 0.01324039
#> -1.363329e+00 0.72633527
#> 1.230571e+00 2.23150783
#> -2.845506e-01 0.16106127
#> -1.363329e-01 0.07263353
#> 1.230571e-01 0.22315078
#> 1.224420e-03 0.03986666
#> 9.469588e-01 0.99789556
#> 6.592220e-04 0.03469121
#> -1.142476e+00 1.10911957
#> -1.239017e+00 1.40527411
#> -1.142340e+00 1.85676617
#> -1.239017e-01 0.14052741
#> -1.104973e+01 149.46811634
#> 1.345150e-12 0.99988780
#> -9.960108e-01 1.00000000
#> 1.014997e-12 0.99983149
```

We can also plot the estimators bootstrap distributions:

```
plot(extensive_RCT)
```



If we use `bmbstats::RCT_estimators_simple`, we will get much more condensed output:

```

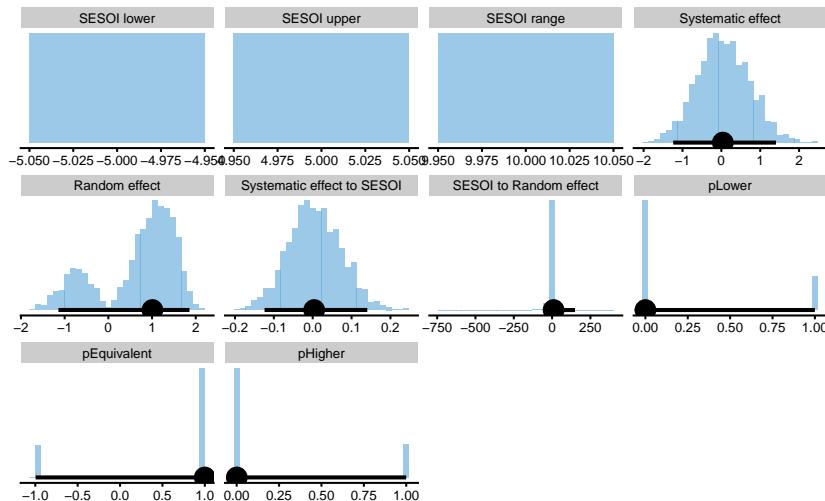
simple_RCT <- bmbstats::RCT_analysis(data = RCT_data, group = "Group",
  treatment_label = "Treatment", control_label = "Control",
  pre_test = "Measured_score.Pre", post_test = "Measured_score.Post",
  SESOI_lower = -5, SESOI_upper = 5, estimator_function = bmbstats::RCT_estimators_simple,
  control = model_control(seed = 1667))
#> [1] "All values of t are equal to 5 \n Cannot calculate confidence intervals"
#> [1] "All values of t are equal to 10 \n Cannot calculate confidence intervals"

simple_RCT
#> Bootstrap with 2000 resamples and 95% bca confidence intervals.
#>
#>           estimator      value      lower
#> SESOI lower -5.0000000000          NA
#> SESOI upper  5.0000000000          NA
#> SESOI range 10.0000000000          NA
#>   Systematic effect  0.0382099536 -1.239017e+00
#>     Random effect  1.0083680171 -1.142340e+00
#> Systematic effect to SESOI 0.0038209954 -1.239017e-01
#>   SESOI to Random effect 9.9170142553 -1.104973e+01
#>           pLower 0.0003713043  1.345150e-12
#>           pEquivalent 0.9992167407 -9.960108e-01
#>           pHigher 0.0004119550  1.014997e-12
#>           upper
#>           NA

```

```
#>      NA
#>      NA
#> 1.4052741
#> 1.8567662
#> 0.1405274
#> 149.4681163
#> 0.9998878
#> 1.0000000
#> 0.9998315
```

```
plot(simple_RCT)
```

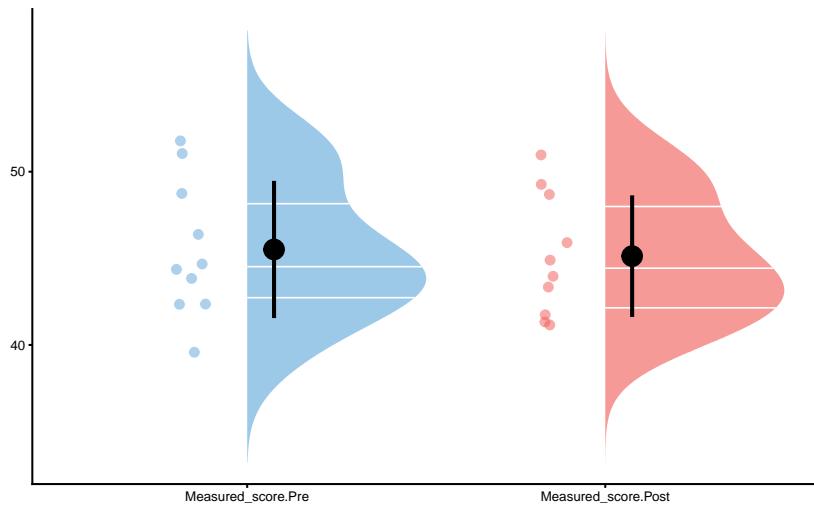


As can be seen, the analysis correctly identified no treatment effect. There is an issue with random treatment effects estimation since it demonstrates distribution with two peaks. This effect is due to random treatment effect being zero and the way the root of the squared differences is calculated to avoid irrational numbers (i.e. taking root of negative number).

There are additional graphs that can be produced. All graphs can take `control = plot_control` parameter to setup plotting options as explained in the previous chapters.

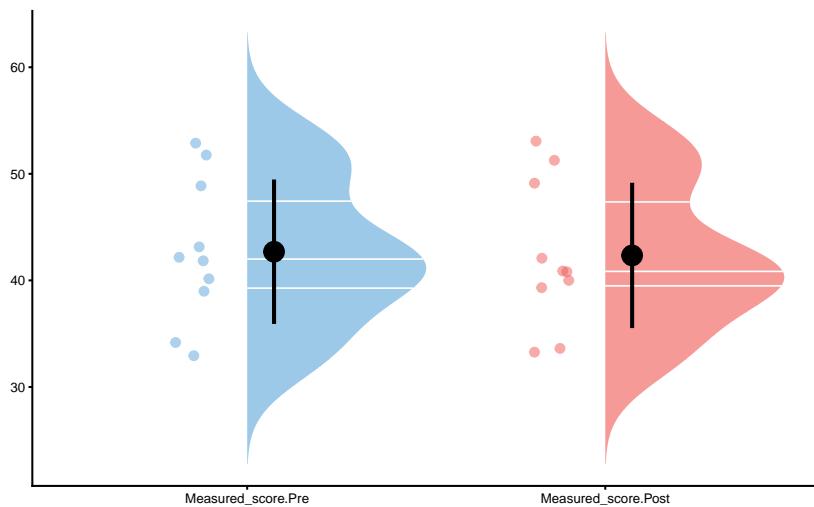
Control group Pre- and Post-test distribution:

```
plot(simple_RCT, type = "control-pre-post")
```



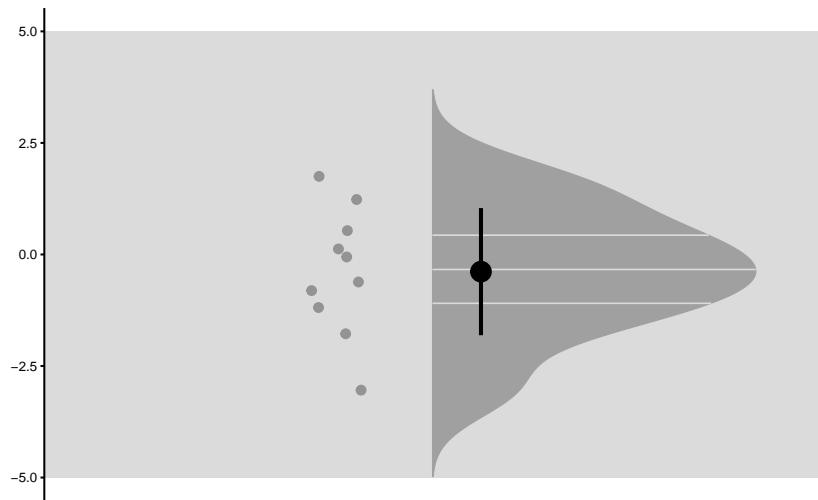
Treatment group Pre- and Post-test distribution:

```
plot(simple_RCT, type = "treatment-pre-post")
```



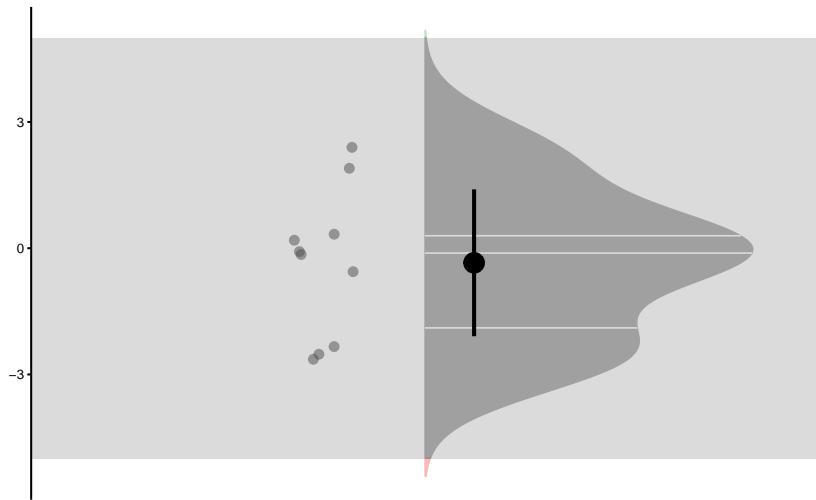
Control change graph:

```
plot(simple_RCT, type = "control-change")
```



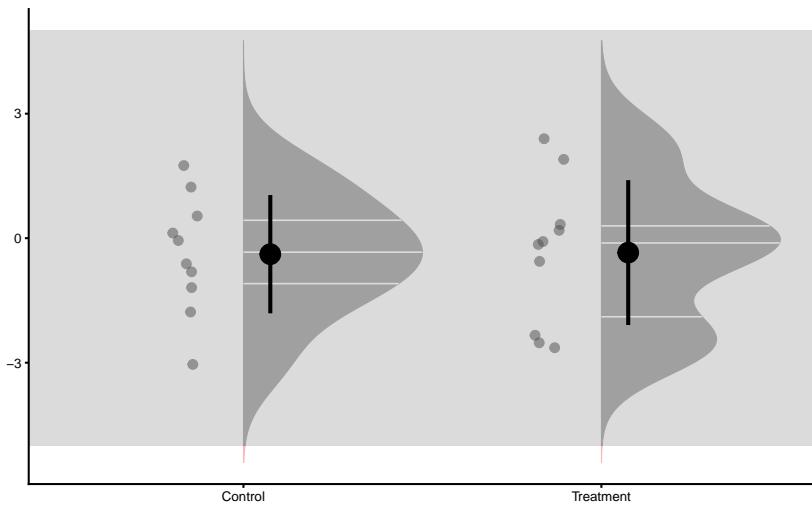
Treatment change graph:

```
plot(simple_RCT, type = "treatment-change")
```



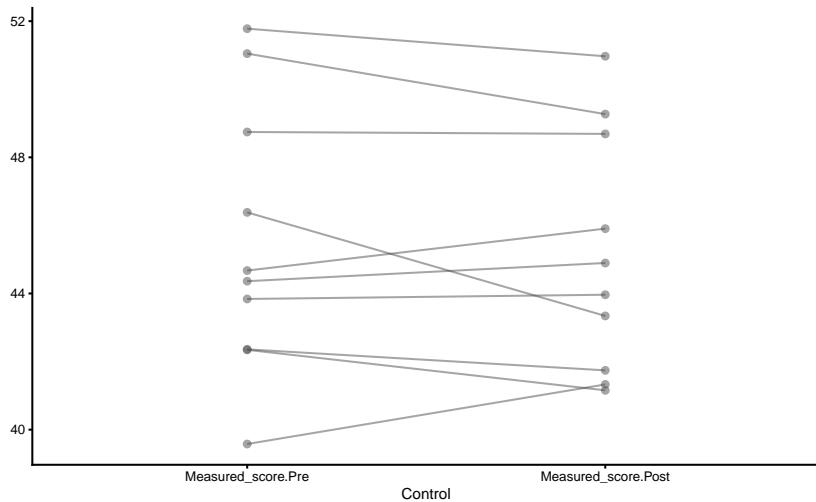
Change graph:

```
plot(simple_RCT, type = "change")
```



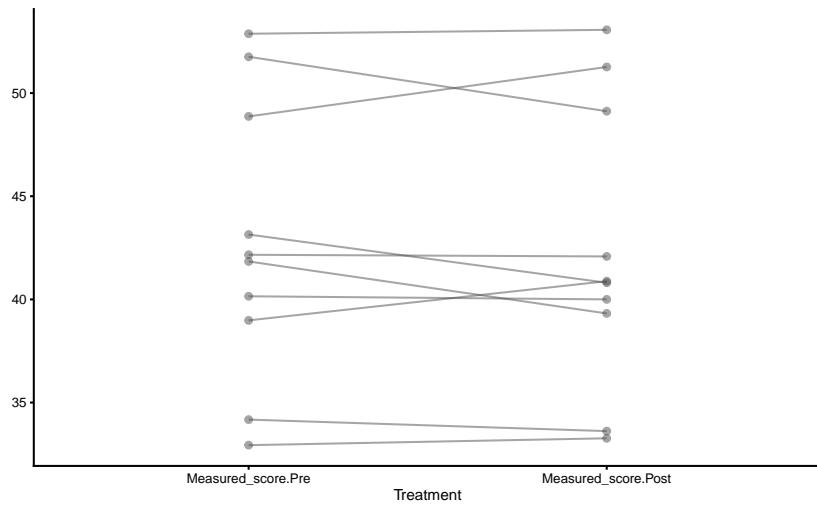
Individual changes in the Control group:

```
plot(simple_RCT, type = "control-paired-change")
```



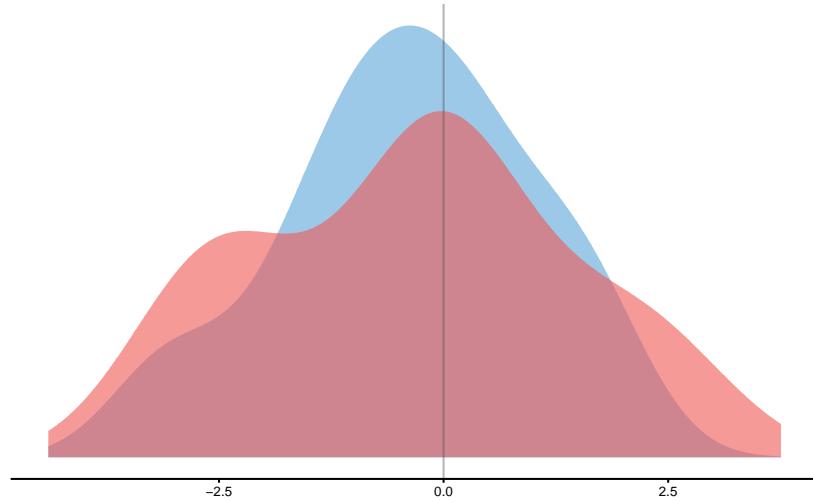
Individual changes in the Treatment group:

```
plot(simple_RCT, type = "treatment-paired-change")
```



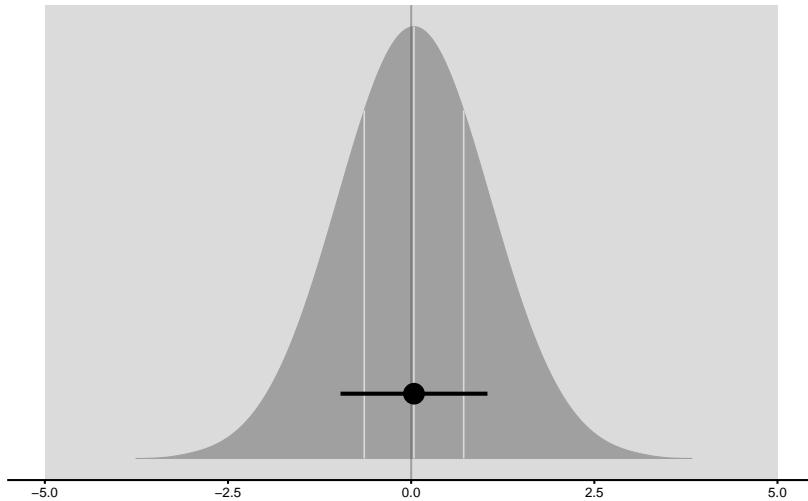
Distribution of the change scores:

```
plot(simple_RCT, type = "change-distribution")
```



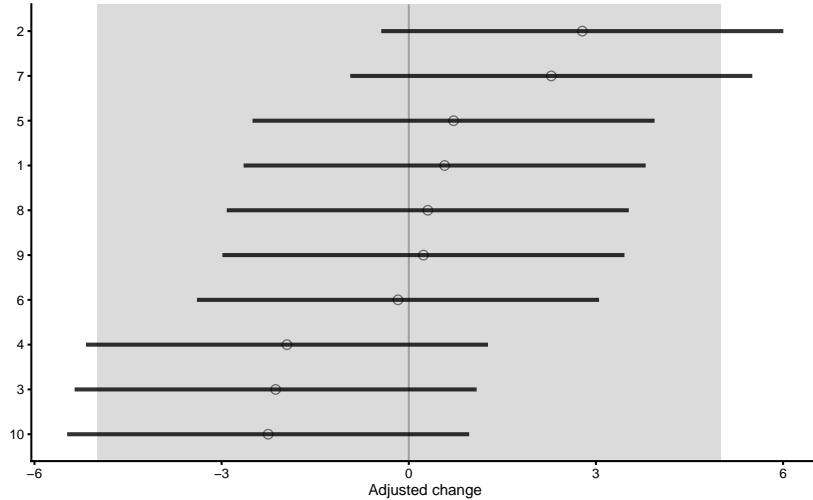
Treatment effect distribution:

```
plot(simple_RCT, type = "effect-distribution")
```



And finally, adjusted treatment responses:

```
plot(simple_RCT, type = "adjusted-treatment-responses")
```



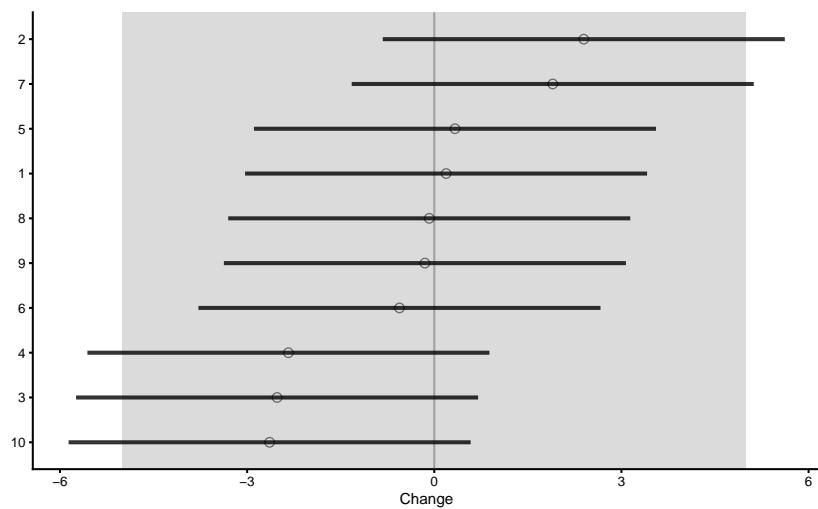
The adjusted treatment responses are calculated by deducting `mean` Control group change from individual change in the Treatment group (i.e. *adjusted change*). Error-bars represent 95% confidence intervals (i.e. SDC) using change SD of the Control group.

The data used to create this graph can be found in the returned object:

```
head(simple_RCT$extra$treatment_responses)
#>   id      group pre_test post_test      change      SDC
#> 11 1 Treatment 52.87665 53.06613  0.1894841 3.222294
#> 12 2 Treatment 48.86917 51.26538  2.3962116 3.222294
#> 13 3 Treatment 41.83908 39.31947 -2.5196029 3.222294
#> 14 4 Treatment 43.14467 40.80646 -2.3382079 3.222294
#> 15 5 Treatment 32.93471 33.26669  0.3319806 3.222294
#> 16 6 Treatment 34.17358 33.61544 -0.5581425 3.222294
#>   change_lower change_upper adjusted_change
#> 11    -3.0328104    3.4117785     0.5746587
#> 12    -0.8260828    5.6185061     2.7813863
#> 13    -5.7418974    0.7026915    -2.1344283
#> 14    -5.5605023    0.8840866    -1.9530332
#> 15    -2.8903138    3.5542751     0.7171553
#> 16    -3.7804369    2.6641520    -0.1729678
#>   adjusted_change_lower adjusted_change_upper
#> 11          -2.6476357        3.796953
#> 12          -0.4409082        6.003681
#> 13          -5.3567227        1.087866
#> 14          -5.1753277        1.269261
#> 15          -2.5051392        3.939450
#> 16          -3.3952623        3.049327
```

We can also plot the un-adjusted treatment responses:

```
plot(simple_RCT, type = "treatment-responses")
```



Let's re-create this graph using `bmbstats::observations_MET` function since that function also allows us to set Type I error rates and confidence for plotting.

```
treatment_group <- filter(
  RCT_data,
  Group == "Treatment"
)

control_group <- filter(
  RCT_data,
  Group == "Control"
)

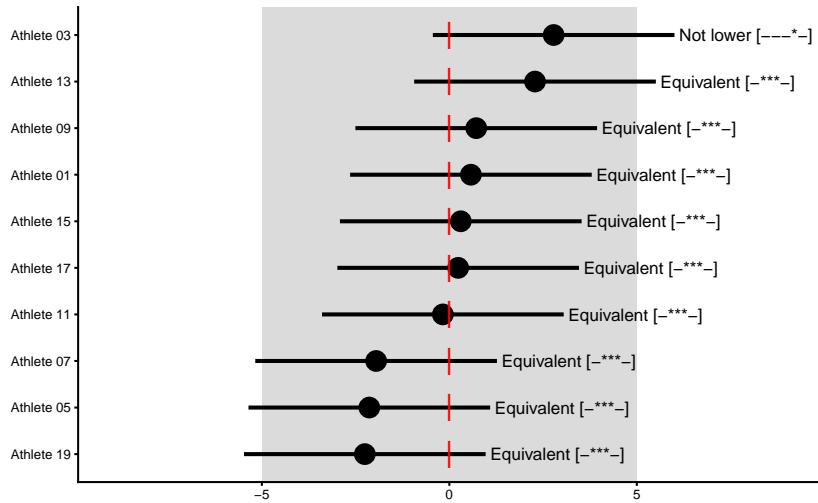
treatment_responses <- bmbstats::observations_MET(
  observations = treatment_group$Measured_score.Change - mean(control_group$Measured_score.Change),
  observations_label = treatment_group$Athlete,

  # Use control group change as measurement error
  measurement_error = sd(control_group$Measured_score.Change),

  # Degrees of freedom from the reliability study. Use `Inf` for normal distribution
  df = nrow(control_group) - 1,
  SESOI_lower = -5,
  SESOI_upper = 5,

  # Will not use Bonferroni adjustment here
  alpha = 0.05,
  # No adjustment in CIs for plotting
  confidence = 0.95
)

plot(
  treatment_responses,
  true_observations = treatment_group$True_score.Change,
  control = plot_control(points_size = 5)
) +
  xlim(-9, 9)
```

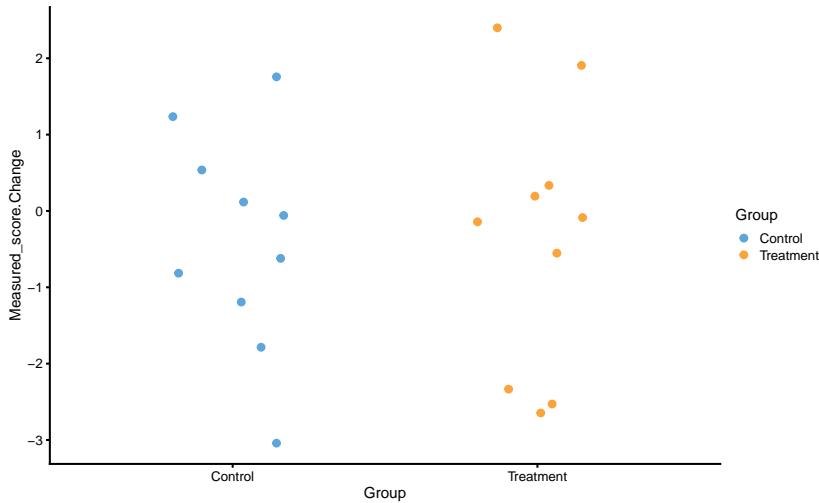


This way, Control group is used as *sort-of* reliability study (see [Repeatability](#) section in the [Validity and Reliability](#) chapter) that provides source of information about the non-treatment effect (in this case 0 for both systematic and random components), biological variation and instrumentation noise. This helps us to provide uncertainty intervals around individual treatment (adjusted) effects.

16.3 Linear Regression Perspective

RCT analysis performed with `bmbstats::RCT_analysis` is the method of differences (i.e. changes). This is similar to the method of differences explained in the [Validity and Reliability](#) chapter. Another approach is to utilize linear regression (explained in the [Prediction as a complement to causal inference](#) section of the [Causal inference](#) chapter). To understand this, let's depict this RCT data by using Group as predictor and Change score as outcome:

```
ggplot(RCT_data, aes(x = Group, y = Measured_score.Change,
color = Group)) + theme_cowplot(8) + geom_jitter(width = 0.2) +
scale_color_manual(values = c(Treatment = "#FAA43A",
Control = "#5DA5DA"))
```



If we perform the linear regression, we will get the following:

```
model1 <- lm(Measured_score.Change ~ Group, RCT_data)

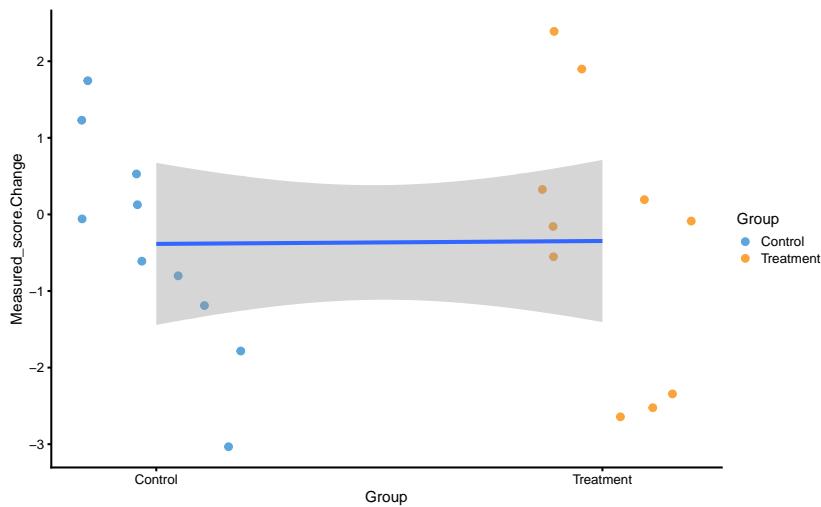
summary(model1)
#>
#> Call:
#> lm(formula = Measured_score.Change ~ Group, data = RCT_data)
#>
#> Residuals:
#>      Min       1Q   Median       3Q      Max 
#> -2.6559 -0.9517  0.2315  0.7391  2.7432 
#>
#> Coefficients:
#>             Estimate Std. Error t value Pr(>|t|)    
#> (Intercept) -0.38517   0.50373 -0.765   0.454    
#> GroupTreatment 0.03821   0.71238  0.054   0.958    
#>
#> Residual standard error: 1.593 on 18 degrees of freedom
#> Multiple R-squared:  0.0001598, Adjusted R-squared:  -0.05539 
#> F-statistic: 0.002877 on 1 and 18 DF,  p-value: 0.9578
```

Intercept in this case (-0.39cm) represents the mean change in the Control group, while slope (GroupTreatment; 0.04cm) represents estimate of the systematic treatment effect (i.e. difference in means).

This can be easily explained visually:

```
RCT_data$Group_num <- ifelse(RCT_data$Group == "Treatment",
  1, 0)

ggplot(RCT_data, aes(x = Group_num, y = Measured_score.Change)) +
  theme_cowplot(8) + geom_jitter(aes(color = Group), width = 0.2) +
  geom_smooth(method = "lm", formula = y ~ x, se = TRUE) +
  scale_color_manual(values = c(Treatment = "#FAA43A",
    Control = "#5DA5DA")) + scale_x_continuous(name = "Group",
  breaks = c(0, 1), labels = c("Control", "Treatment"))
```



Random components of the treatment and non-treatment effects can be seen as *residuals* (actually as SD of the residuals) around *mean* of the group changes. RSE of this model (1.59cm) represents *pooled* random errors of both Treatment and Control groups. If we perform SD of the residuals for each Group, we will get the following:

```
SD_summary <- RCT_data %>% mutate(.resid = residuals(model1)) %>%
  group_by(Group) %>% summarise(`Residual SD` = sd(.resid))

SD_summary
#> # A tibble: 2 x 2
#>   Group      `Residual SD`
#>   <fct>        <dbl>
#> 1 Control      1.42
#> 2 Treatment    1.75
```

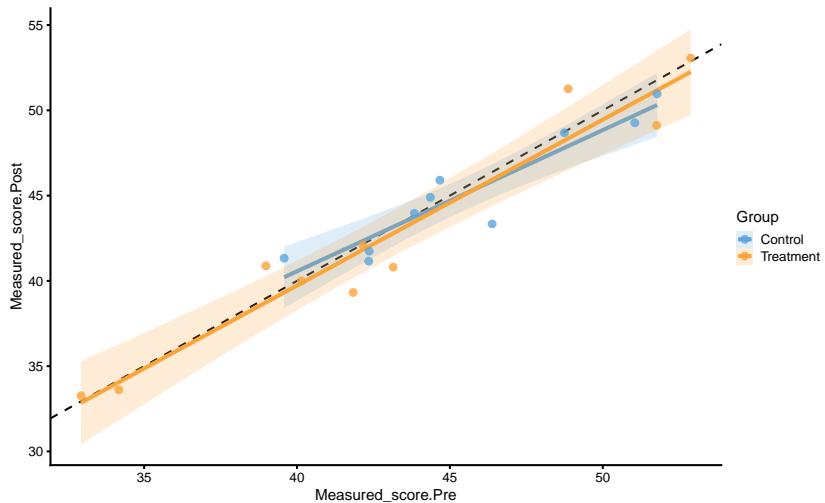
Estimating random treatment effect is thus equal to $\sqrt{SD_{Treatment\ group}^2 - SD_{Control\ group}^2}$:

```
sqrt(SD_summary$`Residual SD`[2]^2 - SD_summary$`Residual SD`[1]^2)
#> [1] 1.008368
```

If you compare these regression estimates to those from the `bmbstats::RCT_analysis`, you will notice they are equal.

But, as alluded in [Causal inference](#) chapter, one should avoid using Change scores and rather use Pre-test and Post-test scores. In this case, our model would look visually like this (I have added dashed identity line so Pre-test and Post-test comparison is easier):

```
ggplot(RCT_data, aes(x = Measured_score.Pre, y = Measured_score.Post,
  color = Group, fill = Group)) + theme_cowplot(8) + geom_abline(slope = 1,
  linetype = "dashed") + geom_smooth(method = "lm", se = TRUE,
  alpha = 0.2) + geom_point(alpha = 0.8) + scale_color_manual(values = c(Treatment = "#FAA43A",
  Control = "#5DA5DA")) + scale_fill_manual(values = c(Treatment = "#FAA43A",
  Control = "#5DA5DA"))
```



And estimated linear regression model:

```
model2 <- lm(Measured_score.Post ~ Measured_score.Pre + Group,
  RCT_data)

summary(model2)
#>
#> Call:
#> lm(formula = Measured_score.Post ~ Measured_score.Pre + Group,
```

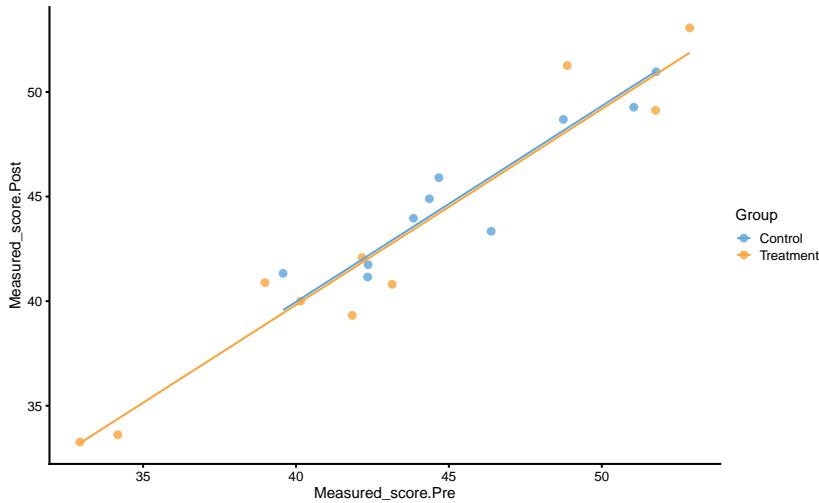
```
#>      data = RCT_data)
#>
#> Residuals:
#>    Min     1Q Median     3Q    Max
#> -2.60014 -1.01512  0.04426  0.93174  3.13880
#>
#> Coefficients:
#>              Estimate Std. Error t value Pr(>|t|)
#> (Intercept) 2.52854   3.12937   0.808   0.43
#> Measured_score.Pre 0.93598   0.06786  13.793 1.16e-10
#> GroupTreatment -0.14242   0.73977  -0.193   0.85
#>
#> (Intercept)
#> Measured_score.Pre ***
#> GroupTreatment
#> ---
#> Signif. codes:
#> 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 1.598 on 17 degrees of freedom
#> Multiple R-squared:  0.9236, Adjusted R-squared:  0.9146
#> F-statistic: 102.7 on 2 and 17 DF,  p-value: 3.22e-10
```

In this model, Pre-test represent covariate. The estimated coefficient for this covariate is 0.94, or very close to 1.

The systematic treatment effect (-0.14cm) can be interpreted as “everything else being equal, changing group from Control to Treatment will result in” effect on the Post-test. Graphically, this represent the *vertical gap* between the two lines. Please note that in the previous graph, each group is modeled separately, and in this model the lines are assumed to be parallel (i.e. no interaction between the lines - or no interaction between Group and Pre-test). Let us thus plot our model:

```
RCT_data$.pred <- predict(model2, newdata = RCT_data)

ggplot(RCT_data, aes(x = Measured_score.Pre, y = Measured_score.Post,
color = Group, fill = Group)) + theme_cowplot(8) + geom_point(alpha = 0.8) +
geom_line(aes(y = .pred)) + scale_color_manual(values = c(Treatment = "#FAA43A",
Control = "#5DA5DA")) + scale_fill_manual(values = c(Treatment = "#FAA43A",
Control = "#5DA5DA"))
```



As can be seen from the figure, lines are almost identical, but more importantly, they are now parallel.

To estimate random treatment effect, we can use residuals SD for each group yet again.

```
SD_summary <- RCT_data %>% mutate(.resid = residuals(model2)) %>%
  group_by(Group) %>% summarise(`Residual SD` = sd(.resid))

SD_summary
#> # A tibble: 2 x 2
#>   Group      `Residual SD`
#>   <fct>     <dbl>
#> 1 Control    1.32
#> 2 Treatment  1.75

sqrt(SD_summary$`Residual SD`[2]^2 - SD_summary$`Residual SD`[1]^2)
#> [1] 1.151716
```

This concept was a bit hard for me to wrap my head around as well. Residuals are something we cannot explain with our model (be it using simple `mean` with the method of differences, or using linear regression model with Group and Pre-test). Everything we can explain with the model can be thus seen as systematic effect. Thus, the *thing that is left* is unexplained variance. This variance is due to biological variation, instrumentation error and random non-treatment effects (which are zero in this example), as well as random treatment effects (which are also zero in this example).

This approach is implemented in `bmbstats::RCT_estimators_lm` function that we can use in `bmbstats::RCT_analysis` to get bootstrap CIs:

```

regression_RCT <- bmbstats::RCT_analysis(data = RCT_data,
  group = "Group", treatment_label = "Treatment", control_label = "Control",
  pre_test = "Measured_score.Pre", post_test = "Measured_score.Post",
  SESOI_lower = -5, SESOI_upper = 5, estimator_function = bmbstats::RCT_estimators_l1,
  control = model_control(seed = 1667))
#> [1] "All values of t are equal to 5 \n Cannot calculate confidence intervals"
#> [1] "All values of t are equal to 10 \n Cannot calculate confidence intervals"

regression_RCT
#> Bootstrap with 2000 resamples and 95% bca confidence intervals.
#>
#>           estimator      value      lower
#> SESOI lower -5.0000000000          NA
#> SESOI upper 5.0000000000          NA
#> SESOI range 10.0000000000          NA
#>   Systematic effect -0.1424169958 -1.545188e+00
#>       Random effect  1.1517164871 -8.277225e-01
#> Systematic effect to SESOI -0.0142416996 -1.545188e-01
#>   SESOI to Random effect  8.6826924091 -2.122197e+01
#>           pLower  0.0002330614  1.379717e-14
#>           pEquivalent 0.9996342091 -9.999971e-01
#>           pHigher  0.0001327295  1.787965e-15
#>           upper
#>           NA
#>           NA
#>           NA
#> 1.6146987
#> 1.9466856
#> 0.1614699
#> 39.5968395
#> 1.0000000
#> 1.0000000
#> 1.0000000

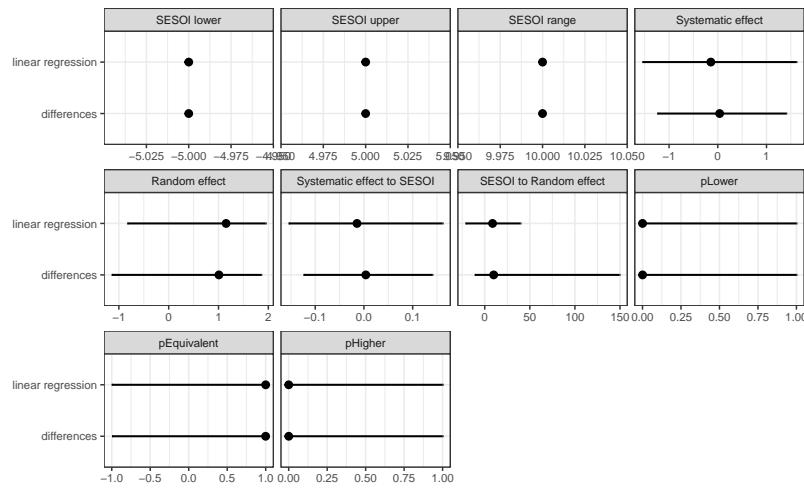
```

Let's now compare the results of the linear regression method with the method of differences (which uses Change score analysis):

```
compare_methods <- rbind(data.frame(method = "differences",
  simple_RCT$estimators), data.frame(method = "linear regression",
  regression_RCT$estimators))

ggplot(compare_methods, aes(y = method, x = value)) + theme_bw(8) +
  geom_errorbarh(aes(xmax = upper, xmin = lower), color = "black",
```

```
height = 0) + geom_point() + xlab("") + ylab("") +
facet_wrap(~estimator, scales = "free_x")
```



Which method should be used? For simple designs, these provide almost identical results. We could perform simulations as we have done in the [Validity and Reliability](#) chapter to see the behavior of the estimates, but I will leave that to you as an exercise. I would generally follow the advice by Frank Harrell¹ and avoid the use of the change scores, particularly for more complex designs involving covariates and extra parameters of the treatment (i.e. when treatment is not only TRUE/FALSE but can have a continuous membership function, or some type of the loading parameter, like number of jumps performed and so forth).

16.4 Prediction perspective 1

Using simple linear regression function brings us to the prediction perspective of modeling the RCT data. Rather than estimating DGP parameters, we are interested in predicting individual responses on the unseen data. We can thus use `bmbstats::cv_model` function to estimate predictive performance of our simple linear regression model. One thing to keep in mind, is that now we are using SESOI to judge the residuals magnitudes, not observed scores (i.e. measured Post-test). Please note that I have used Group column to setup the CV strata - this results in same number of CV observations in each group.

¹<https://www.fharrell.com/post/errmed/#change>

```

model3 <- cv_model(Measured_score.Post ~ Measured_score.Pre +
  Group, RCT_data, SESOI_lower = -5, SESOI_upper = 5, control = model_control(seed =
  cv_folds = 3, cv_repeats = 10, cv_strata = factor(RCT_data$Group)))

model3
#> Training data consists of 3 predictors and 20 observations. Cross-Validation of the
#>
#> Model performance:
#>
#>       metric      training training.pooled
#>       MBE   2.202681e-14   1.918475e-15
#>       MAE   1.176009e+00   1.125490e+00
#>       RMSE  1.473109e+00   1.403919e+00
#>       PPER  9.963039e-01   9.995798e-01
#>   SESOI to RMSE  6.788364e+00   7.122918e+00
#>       R-squared  9.235673e-01   9.305786e-01
#>       MinErr   -3.138804e+00  -3.623411e+00
#>       MaxErr   2.600139e+00   2.890624e+00
#>       MaxAbsErr 3.138804e+00   3.623411e+00
#> testing.pooled      mean          SD          min
#>   -0.04123424 -0.06466644 0.91228266 -2.0716103
#>   1.47603175  1.47790175 0.33700044  0.8701076
#>   1.82619034  1.79036050 0.37125297  1.1312614
#>   0.99312088  0.96151334 0.02661372  0.8929713
#>   5.47588047  5.82060839 1.20914378  3.7299209
#>   0.88259682  0.86160357 0.10805679  0.5866306
#>   -5.03757174 -2.56189890 1.14780901 -5.0375717
#>   3.53991399  2.08918814 0.96885433 -0.4470821
#>   5.03757174  3.08675364 0.76498989  1.7111147
#>           max
#>   1.8762924
#>   2.0982105
#>   2.6810220
#>   0.9939017
#>   8.8396899
#>   0.9743506
#>   0.3235147
#>   3.5399140
#>   5.0375717

```

As can be seen from the performance results, we have pretty good prediction of the individual responses since our SESOI is pretty large and we do not have any treatment nor non-treatment effects.

bmbstats package comes with additional function `bmbstats::RCT_predict` that uses returned object by `bmbstats::cv_model` and analyze is from RCT

perspective:

```

prediction_RCT <- RCT_predict(model3, new_data = RCT_data,
                                outcome = "Measured_score.Post", group = "Group", treatment_label = "Treatment",
                                control_label = "Control", subject_label = RCT_data$Athlete)

prediction_RCT
#> Training data consists of 3 predictors and 20 observations. Cross-Validation of the model was
#>
#> Model performance:
#>
#>           metric      training training.pooled
#>           MBE   2.202681e-14    1.918475e-15
#>           MAE   1.176009e+00    1.125490e+00
#>           RMSE  1.473109e+00    1.403919e+00
#>           PPER  9.963039e-01    9.995798e-01
#> SESOI to RMSE 6.788364e+00    7.122918e+00
#> R-squared 9.235673e-01    9.305786e-01
#> MinErr -3.138804e+00    -3.623411e+00
#> MaxErr  2.600139e+00    2.890624e+00
#> MaxAbsErr 3.138804e+00    3.623411e+00
#> testing.pooled      mean          SD          min
#> -0.04123424 -0.06466644 0.91228266 -2.0716103
#> 1.47603175  1.47790175 0.33700044  0.8701076
#> 1.82619034  1.79036050 0.37125297  1.1312614
#> 0.99312088  0.96151334 0.02661372  0.8929713
#> 5.47588047  5.82060839 1.20914378  3.7299209
#> 0.88259682  0.86160357 0.10805679  0.5866306
#> -5.03757174 -2.56189890 1.14780901 -5.0375717
#> 3.53991399  2.08918814 0.96885433 -0.4470821
#> 5.03757174  3.08675364 0.76498989  1.7111147
#> max
#> 1.8762924
#> 2.0982105
#> 2.6810220
#> 0.9939017
#> 8.8396899
#> 0.9743506
#> 0.3235147
#> 3.5399140
#> 5.0375717
#>
#> Individual model results:
#>
#>     subject      group observed predicted      residual
#> Athlete 01 Treatment 53.06613 51.87749 -1.18864415

```

```

#> Athlete 02 Control 41.74203 42.17581 0.43378581
#> Athlete 03 Treatment 51.26538 48.12658 -3.13880386
#> Athlete 04 Control 44.89588 44.04977 -0.84610959
#> Athlete 05 Treatment 39.31947 41.54657 2.22709299
#> Athlete 06 Control 41.15527 42.16271 1.00744225
#> Athlete 07 Treatment 40.80646 42.76857 1.96211098
#> Athlete 08 Control 48.68788 48.15176 -0.53612109
#> Athlete 09 Treatment 33.26669 33.21228 -0.05441276
#> Athlete 10 Control 45.90508 44.34223 -1.56285495
#> Athlete 11 Treatment 33.61544 34.37183 0.75639501
#> Athlete 12 Control 43.34150 45.94164 2.60013908
#> Athlete 13 Treatment 40.88270 38.87404 -2.00866163
#> Athlete 14 Control 50.96896 50.99231 0.02334441
#> Athlete 15 Treatment 42.08253 41.84993 -0.23260068
#> Athlete 16 Control 49.26936 50.30751 1.03815067
#> Athlete 17 Treatment 40.00091 39.96681 -0.03410111
#> Athlete 18 Control 41.32829 39.57284 -1.75545843
#> Athlete 19 Treatment 49.12101 50.83263 1.71162522
#> Athlete 20 Control 43.96351 43.56119 -0.40231815
#> magnitude counterfactual pITE pITE_magnitude
#> Equivalent 52.01990 0.142417 Equivalent
#> Equivalent 42.03340 -0.142417 Equivalent
#> Equivalent 48.26899 0.142417 Equivalent
#> Equivalent 43.90736 -0.142417 Equivalent
#> Equivalent 41.68898 0.142417 Equivalent
#> Equivalent 42.02030 -0.142417 Equivalent
#> Equivalent 42.91099 0.142417 Equivalent
#> Equivalent 48.00934 -0.142417 Equivalent
#> Equivalent 33.35469 0.142417 Equivalent
#> Equivalent 44.19981 -0.142417 Equivalent
#> Equivalent 34.51425 0.142417 Equivalent
#> Equivalent 45.79923 -0.142417 Equivalent
#> Equivalent 39.01645 0.142417 Equivalent
#> Equivalent 50.84989 -0.142417 Equivalent
#> Equivalent 41.99235 0.142417 Equivalent
#> Equivalent 50.16510 -0.142417 Equivalent
#> Equivalent 40.10923 0.142417 Equivalent
#> Equivalent 39.43042 -0.142417 Equivalent
#> Equivalent 50.97505 0.142417 Equivalent
#> Equivalent 43.41878 -0.142417 Equivalent
#>
#> Summary of residuals per RCT group:
#>
#>      group        mean         SD
#> Control 1.634246e-14 1.322097

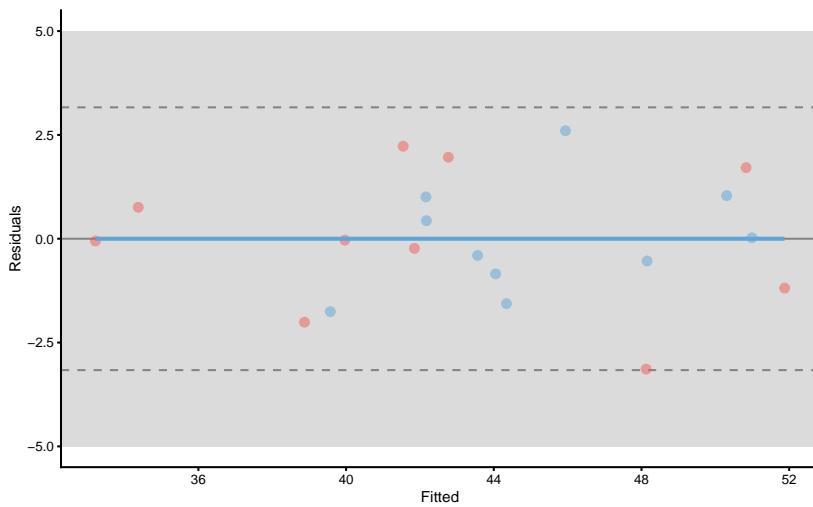
```

```
#> Treatment 2.771120e-14 1.753394
#>
#> Summary of counterfactual effects of RCT group:
#>
#>      group      pATE pVTE
#> Treatment  0.142417   0
#> Control    -0.142417   0
#> pooled     0.142417   0
#>
#> Treatment effect summary
#>
#> Average Treatment effect: 0.142417
#> Variable Treatment effect: 0
#> Random Treatment effect: 1.151716
```

The results of `bmbstats::RCT_predict` contain (1) model performance (also returned from `bmbstats::cv_model`), (2) individual model results (which contains individual model predictions, as well as counterfactual predictions assuming Group changes from Control to Treatment and *vice versa*), (3) residuals summary per group (these are our random effects), and (4) summary of counterfactual effects (please refer to [Causal inference](#) chapter for more information about these concepts). The `extra` details that `bmbstats::RCT_predict` adds on top of `bmbstats::cv_model` can be found in the `prediction_RCT$extra`.

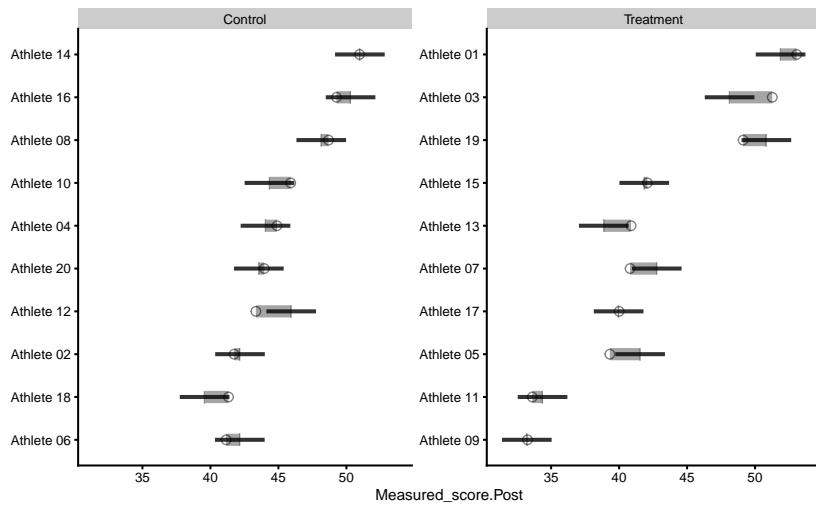
Now that we have the more info about the underlying RCT, we can do various plots. Let's plot the residuals first:

```
plot(prediction_RCT, "residuals")
```



To plot individual model predictions use the following:

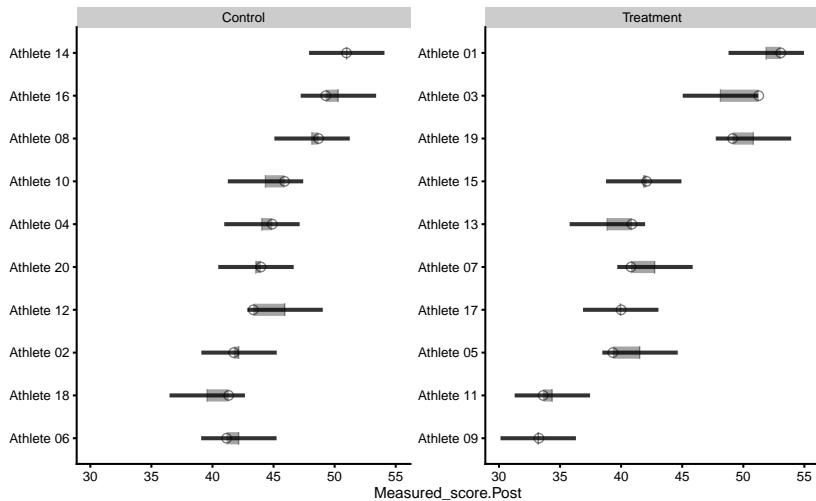
```
plot(prediction_RCT, "prediction")
```



Circlets or dots represent observed (i.e. outcome or target variable), and vertical line represent model predictions. Residual between the two is color coded based on the provided SESOI threshold.

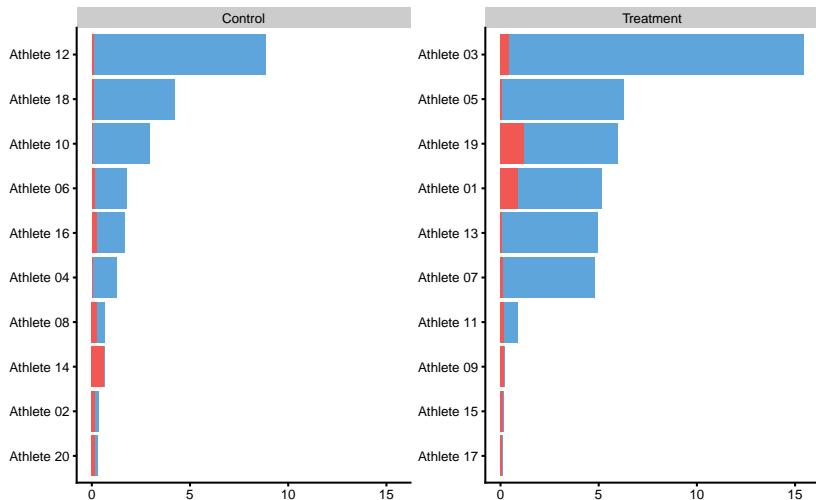
By default, prediction plot uses `metric = "RMSE"` and `metric_cv = "testing.pooled"` parameters to plot confidence intervals. We can use some other metric returned from `bmbstats::cv_model`. Let's use `MaxAbsErr` (maximum absolute prediction error) from mean testing CV column:

```
plot(prediction_RCT, "prediction", metric = "MaxAbsErr",
      metric_cv = "mean")
```



To plot individual bias-variance error decomposition thorough testing CV folds use:

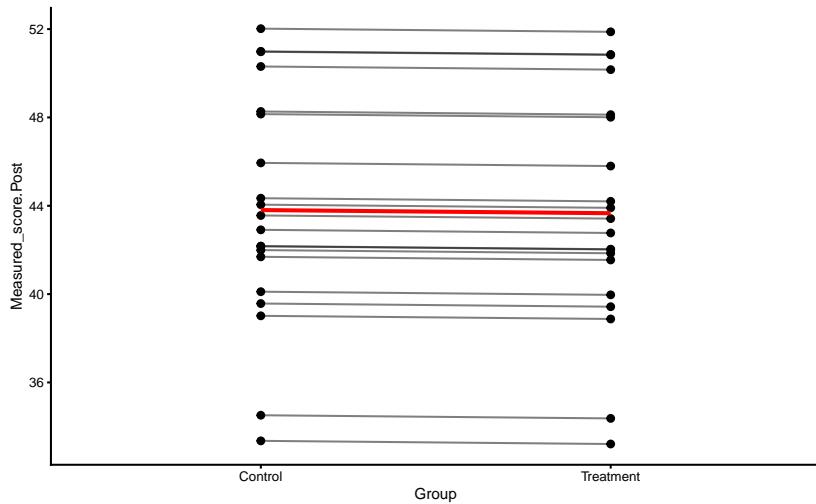
```
plot(prediction_RCT, "bias-variance")
```



Here we can see which athletes are prediction outliers and generally present issues for the predictive model. Together with the individual predictions, we can use this plot and data to gain more info regarding individual reactions to intervention (i.e. who jumps out from the model prediction).

To gain understanding into counterfactual prediction, we could use PDP and ICE plots. The default variable to be used is the Group, which we can change later:

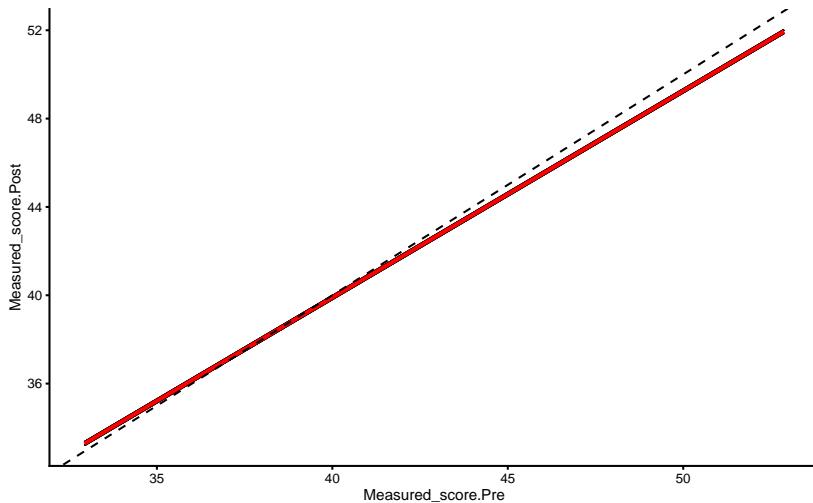
```
plot(prediction_RCT, "pdp+ice")
```



This plot gives us insights into model prediction for each athlete when the Group variable changes, while keeping all other variables the same. Since this is RCT, this plot can be given counterfactual interpretation. The PDP line (thick red line) represent the average of these individual prediction (ICE lines), and as can be seen, the counterfactual effects of changing group is zero (since the line is parallel) and thus represents *expected* or systematic effect of the treatment.

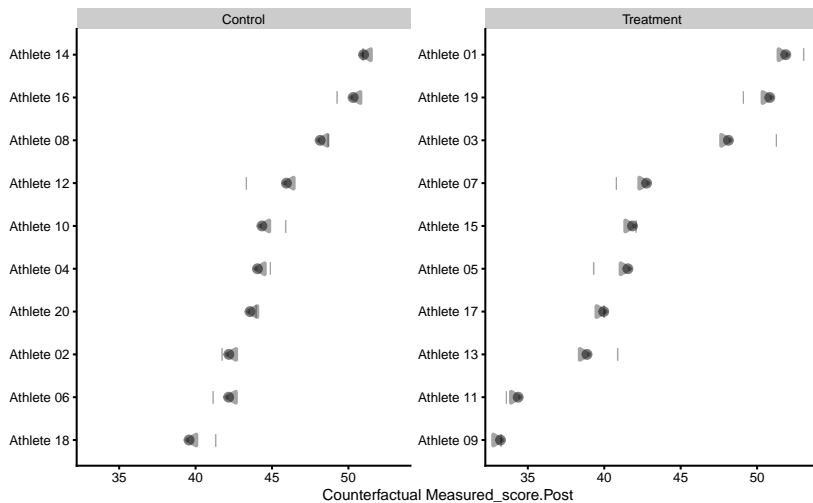
Let's do that for out other variable in the model (with addition of the dashed identity line):

```
plot(prediction_RCT, "pdp+ice", predictor = "Measured_score.Pre") +
  geom_abline(slope = 1, linetype = "dashed")
```



To plot these individual ICE line for each athlete we can use counterfactual plot:

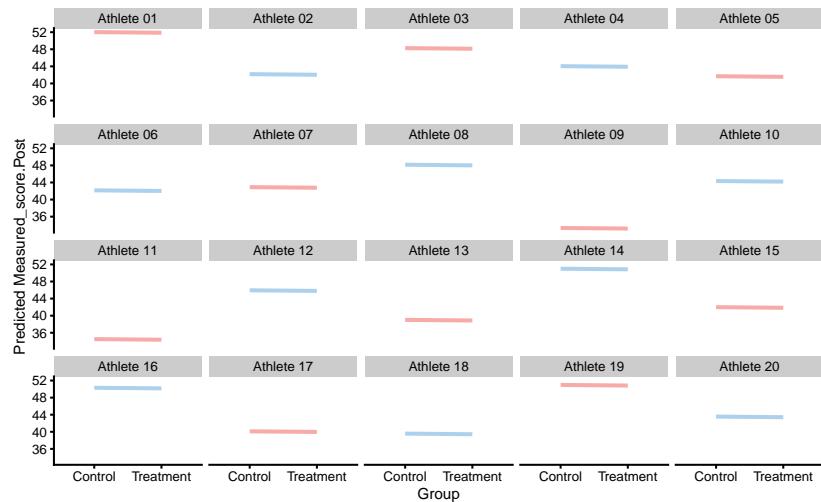
```
plot(prediction_RCT, "counterfactual")
```



Since the systematic treatment effect is zero, these are very small. The mean of these individual counterfactual prediction effect is presented in the object printout in the last table as pATE (predicted average treatment effect) and SD of these effects as pVTE (predicted variable treatment effect). In this case, due simple model, the predicted treatment effects are the same, thus the pVTE is equal to zero.

Different way to plot these is to have *trellis* plot for each individual:

```
plot(prediction_RCT, "ice")
```



These plots represent strong tool for understanding predictive model performance for the RCT data and study designs.

16.5 Adding some effects

The previous example contained no treatment nor non-treatment effects, only biological variation and instrumentation noise. Let's now consider different treatment - plyometric training that has systematic effect of 5cm (`mean` or expected change) and random effect of 5cm. But, since this study is done after the off-season, all athletes experienced systematic effect of 2.5cm increase and random effect of 1cm only by normal training. This represents non-treatment effect.

Biological variation (1.5cm) and instrumentation noise (0.5cm) are still involved in the measured scores.

This is the code to generate the data (i.e. DGP):

```
set.seed(16)

n_subjects <- 20

# These stay the same
instrumentation_noise <- 0.5
biological_variation <- 1.5
```

```

# -----
# Treatment effect
treatment_systematic <- 5
treatment_random <- 5

# Non-treatment effect
non_treatment_systematic <- 2.5
non_treatment_random <- 1
#-----

RCT_data <- tibble(
  Athlete = paste(
    "Athlete",
    str_pad(
      string = seq(1, n_subjects),
      width = 2,
      pad = "0"
    )
  ),
  Group = rep(c("Treatment", "Control"), length.out = n_subjects),

  # True score
  True_score.Pre = rnorm(n_subjects, 45, 5),

  # Treatment effect
  Treatment_effect = rnorm(n = n_subjects, mean = treatment_systematic, sd = treatment_random),
  Non_treatment_effect = rnorm(n = n_subjects, mean = non_treatment_systematic, sd = non_treatment_random)

  # Calculate the change in true score
  True_score.Change = if_else(
    Group == "Treatment",
    # Treatment group is a sum of treatment and non-treatment effects
    Treatment_effect + Non_treatment_effect,
    # While control group only get non-treatment effects
    Non_treatment_effect
  ),
  True_score.Post = True_score.Pre + True_score.Change,

  # Manifested score
  Manifested_score.Pre = True_score.Pre + rnorm(n_subjects, 0, biological_variation),
  Manifested_score.Post = True_score.Post + rnorm(n_subjects, 0, biological_variation),
  Manifested_score.Change = Manifested_score.Post - Manifested_score.Pre,

  # Measured score
  Measured_score.Pre = Manifested_score.Pre + rnorm(n_subjects, 0, instrumentation_noise),

```

```

Measured_score.Post = Manifested_score.Post + rnorm(n_subjects, 0, instrumentation_noise)
Measured_score.Change = Measured_score.Post - Measured_score.Pre
)

RCT_data$Group <- factor(RCT_data$Group)

head(RCT_data)
#> # A tibble: 6 x 13
#>   Athlete Group True_score.Pre Treatment_effect
#>   <chr>    <fct>      <dbl>            <dbl>
#> 1 Athlet~ Trea~        47.4           -3.24
#> 2 Athlet~ Cont~       44.4            3.43
#> 3 Athlet~ Trea~       50.5            4.09
#> 4 Athlet~ Cont~       37.8            12.4
#> 5 Athlet~ Trea~       50.7            0.671
#> 6 Athlet~ Cont~       42.7            12.6
#> # ... with 9 more variables:
#> #   Non_treatment_effect <dbl>,
#> #   True_score.Change <dbl>, True_score.Post <dbl>,
#> #   Manifested_score.Pre <dbl>,
#> #   Manifested_score.Post <dbl>,
#> #   Manifested_score.Change <dbl>,
#> #   Measured_score.Pre <dbl>,
#> #   Measured_score.Post <dbl>,
#> #   Measured_score.Change <dbl>

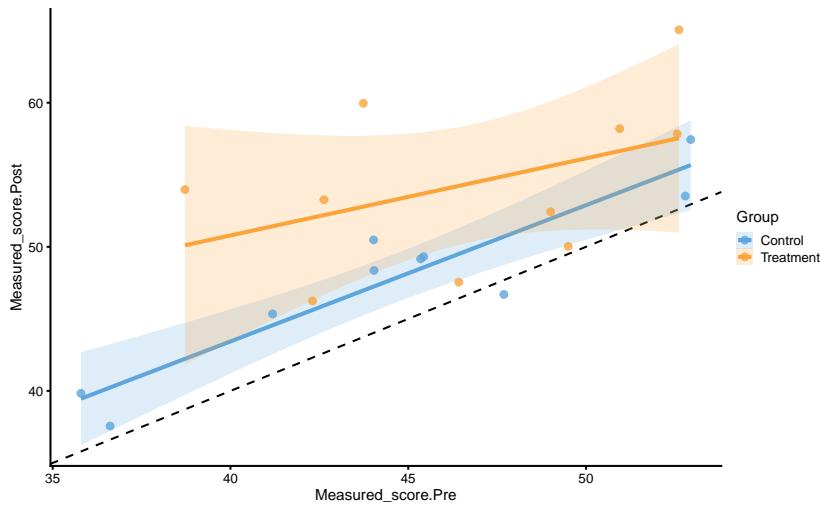
```

Let's plot the data using Pre-test as predictor (x-axis) and Post-test as the outcome (y-axis). Please remember that this plot provide simple linear regression for each group independently. I will add identity line for easier comparison:

```

ggplot(RCT_data, aes(x = Measured_score.Pre, y = Measured_score.Post,
color = Group, fill = Group)) + theme_cowplot(8) + geom_abline(slope = 1,
linetype = "dashed") + geom_smooth(method = "lm", se = TRUE,
alpha = 0.2) + geom_point(alpha = 0.8) + scale_color_manual(values = c(Treatment =
Control = "#5DA5DA")) + scale_fill_manual(values = c(Treatment = "#FAA43A",
Control = "#5DA5DA")))

```



We can see that both Control and Treatment group demonstrated improvements since both lines are above the dashed identity line.

Using simple linear regression we get the following coefficients:

```
model4 <- lm(Measured_score.Post ~ Measured_score.Pre + Group,
               RCT_data)

summary(model4)
#>
#> Call:
#> lm(formula = Measured_score.Post ~ Measured_score.Pre + Group,
#>      data = RCT_data)
#>
#> Residuals:
#>    Min     1Q   Median     3Q    Max
#> -6.5827 -3.5413  0.3856  2.3547  7.9219
#>
#> Coefficients:
#>             Estimate Std. Error t value Pr(>|t|)
#> (Intercept) 13.1094    8.5876  1.527 0.145262
#> Measured_score.Pre 0.7774    0.1902  4.088 0.000767
#> GroupTreatment 4.9265    1.9679  2.503 0.022784
#>
#> (Intercept)
#> Measured_score.Pre ***
#> GroupTreatment *
#> ---
#> Signif. codes:
```

```
#> 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 4.294 on 17 degrees of freedom
#> Multiple R-squared:  0.6289, Adjusted R-squared:  0.5853
#> F-statistic: 14.41 on 2 and 17 DF, p-value: 0.000219
```

Now the simple linear regression estimated systematic treatment effect of 4.93cm, which is very close to our DGP systematic treatment effect of 5cm.

To estimate random treatment effect, we can use residuals SD for each group:

```
SD_summary <- RCT_data %>% mutate(.resid = residuals(model4)) %>%
  group_by(Group) %>% summarise(`Residual SD` = sd(.resid))

SD_summary
#> # A tibble: 2 x 2
#>   Group      `Residual SD`
#>   <fct>        <dbl>
#> 1 Control      2.41
#> 2 Treatment    5.39

sqrt(SD_summary$`Residual SD`[2]^2 - SD_summary$`Residual SD`[1]^2)
#> [1] 4.816519
```

Using `bmbstats::RCT_estimators_lm` estimator function and `bmbstats::RCT_analysis` functions, let's perform the analysis and generate 95% bootstrap confidence intervals:

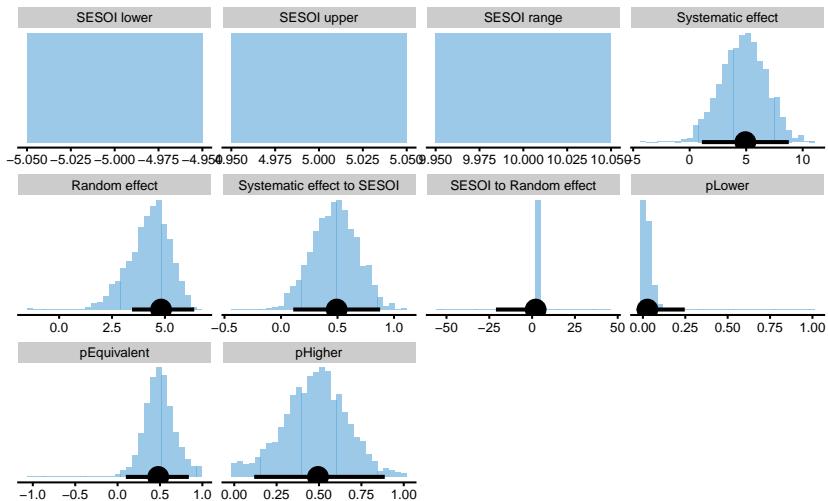
```
regression_RCT <- bmbstats::RCT_analysis(data = RCT_data,
  group = "Group", treatment_label = "Treatment", control_label = "Control",
  pre_test = "Measured_score.Pre", post_test = "Measured_score.Post",
  SESOI_lower = -5, SESOI_upper = 5, estimator_function = bmbstats::RCT_estimators_lm,
  control = model_control(seed = 1667))
#> [1] "All values of t are equal to 5 \n Cannot calculate confidence intervals"
#> [1] "All values of t are equal to 10 \n Cannot calculate confidence intervals"

regression_RCT
#> Bootstrap with 2000 resamples and 95% bca confidence intervals.
#>
#>           estimator      value       lower
#> SESOI lower -5.000000000          NA
#> SESOI upper  5.000000000          NA
#> SESOI range   10.000000000         NA
#> Systematic effect  4.92654585  1.082956784
```

```
#>           Random effect 4.81651919  3.443713578
#> Systematic effect to SESOI 0.49265458  0.108295678
#>      SESOI to Random effect 2.07618814 -21.118934945
#>                  pLower 0.02663292  0.001106225
#>                 pEquivalent 0.47937140  0.097391976
#>                  pHigher 0.49399568  0.117750794
#>      upper
#>          NA
#>          NA
#>          NA
#> 8.7715046
#> 6.3838094
#> 0.8771505
#> 2.9121218
#> 0.2469677
#> 0.8405754
#> 0.8878213
```

Here is the estimators bootstrap distribution:

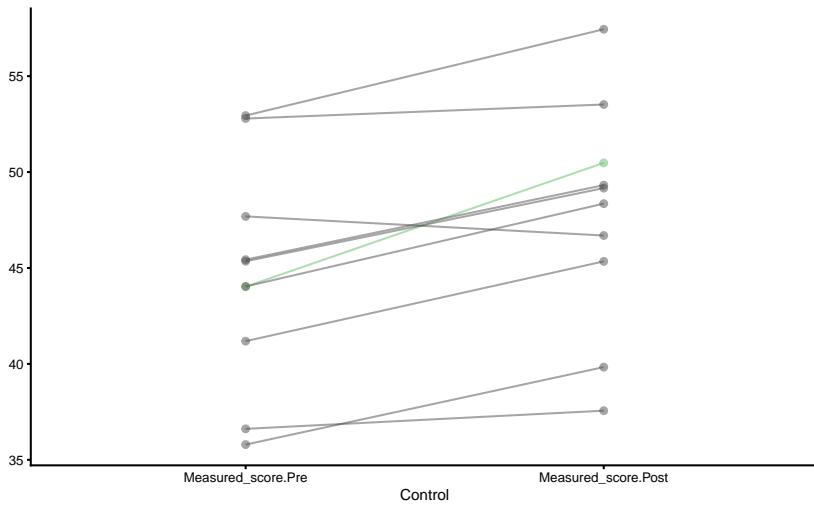
```
plot(regression_RCT)
```



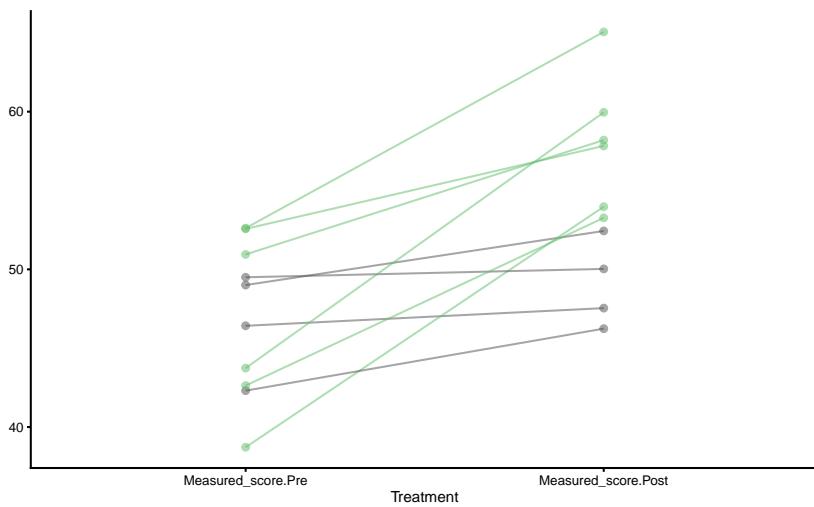
As can be seen from the results, both systematic and random components of the treatment effects were estimated correctly.

Let's plot the Pre- and Post-Test results for each group (I will let you play and plot other figures as explained in the previous sections):

```
plot(regression_RCT, type = "control-paired-change")
```

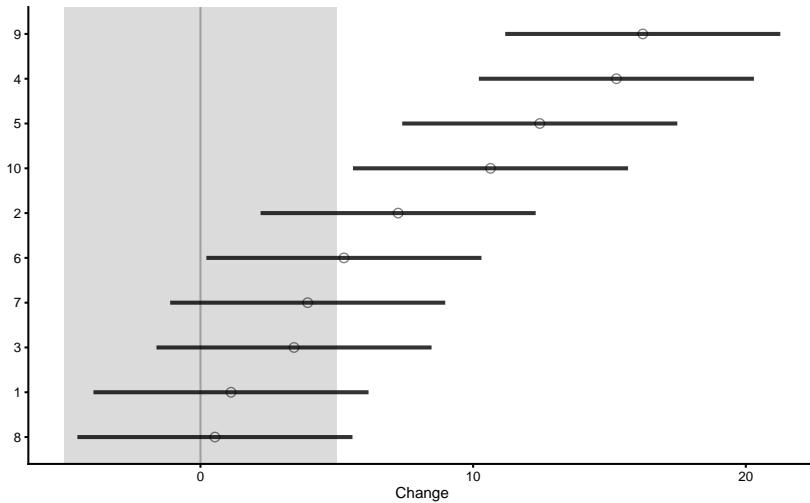


```
plot(regression_RCT, type = "treatment-paired-change")
```



Let's now plot individual treatment responses with uncertainty intervals:

```
plot(regression_RCT, type = "treatment-responses")
```



As explained previously, this type of analysis and plot, uses Control group change score as some type of a *proxy* to quantify the uncertainty around observed treatment group change scores. Please note that even though we have used linear regression approach to estimate treatment effects, the plots still rely on the change scores.

To check if this plot make sense, we can add the true treatment effects from the DGP using `bmbstats::observations_MET` function (we can also use `bmbstats::observations_MBI` function for plotting):

```
treatment_group <- filter(
  RCT_data,
  Group == "Treatment"
)

control_group <- filter(
  RCT_data,
  Group == "Control"
)

treatment_responses <- bmbstats::observations_MET(
  observations = treatment_group$Measured_score.Change,
  observations_label = treatment_group$Athlete,

  # Use control group change as measurement error
  measurement_error = sd(control_group$Measured_score.Change),

  # Degrees of freedom from the reliability study. Use `Inf` for normal distribution
  df = nrow(control_group) - 1,
```

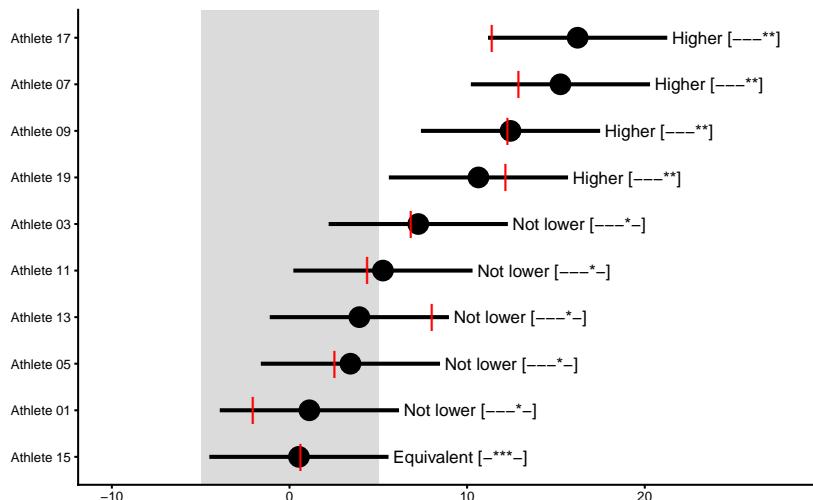
```

SESOI_lower = -5,
SESOI_upper = 5,

# Will not use Bonferroni adjustment here
alpha = 0.05,
# No adjustment in CIs for plotting
confidence = 0.95
)

plot(
  treatment_responses,
  true_observations = treatment_group$True_score.Change,
  control = plot_control(points_size = 5)
) +
  xlim(-10, 28)

```



Adjusted treatment response would deduct `mean` change from the Control group and would show individual effect ONLY with treatment effect (without non-treatment effect). We can use our DGP generated data frame to generate that graph (or to recreate it) and plot true treatment effects:

```

treatment_responses <- bmbstats::observations_MET(
  # Adjustment
  observations = treatment_group$Measured_score.Change - mean(control_group$Measured_s
  observations_label = treatment_group$Athlete,

```

```

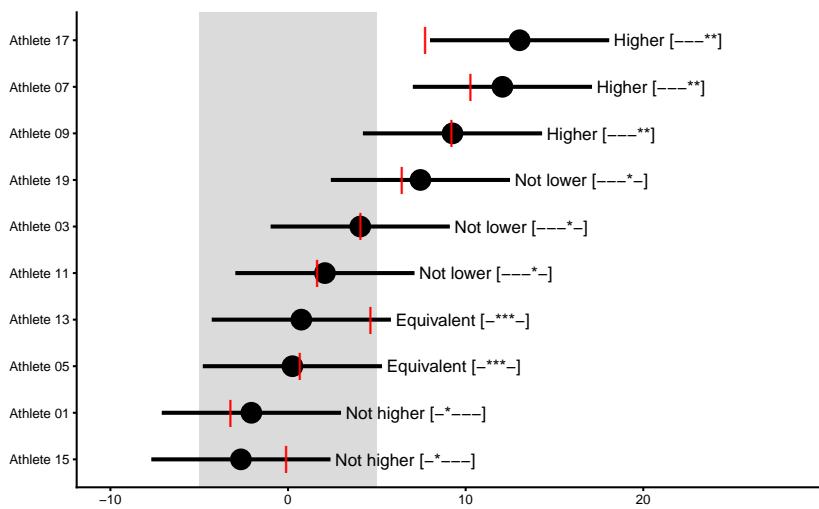
# Use control group change as measurement error
measurement_error = sd(control_group$Measured_score.Change),

# Degrees of freedom from the reliability study. Use `Inf` for normal distribution
df = nrow(control_group) - 1,
SESOI_lower = -5,
SESOI_upper = 5,

# Will not use Bonferroni adjustment here
alpha = 0.05,
# No adjustment in CIs for plotting
confidence = 0.95
)

plot(
treatment_responses,
# The true observation now is the DGP Treatment effect only
true_observations = treatment_group$Treatment_effect,
control = plot_control(points_size = 5)
) +
xlim(-10, 28)

```



16.6 What goes inside the *measurement error* (or Control group change or residuals SD)?

As already explained, Control group serves as a counter-factual proxy of what would happen to the treatment group if not-receiving the treatment. SD of Control group change is used to represent an estimate of uncertainty around individual responses. In our RCT data, SD of the Control group measured change is equal to 2.23cm. This is equal to the root of squared sums of non-treatment random effect, biological variation and instrumentation noise. Since biological variation and instrumentation affects the change score twice (at Pre- and Post-test) we get the following:

$$SD_{control\ change} = \sqrt{2 \times biological\ variation^2 + 2 \times instrumentation\ noise^2 + non\ treatment\ random\ effect^2} \quad (16.1)$$

If we plug our DGP values, we get the following expected SD of the Control group change: 2.45cm. If there is no non-treatment random effect, then SD for the control group would only consist of measurement error.

This measurement error decomposition belongs to the method of differences, but the same error decomposition happens in the SD of the residuals with the linear regression approach.

To demonstrate this, consider random treatment effect estimated using the measured Pre-test and Post-test scores, versus using True scores.

```
model_true <- lm(True_score.Post ~ True_score.Pre + Group,
                   RCT_data)

summary(model_true)
#>
#> Call:
#> lm(formula = True_score.Post ~ True_score.Pre + Group, data = RCT_data)
#>
#> Residuals:
#>    Min      1Q Median      3Q     Max
#> -8.945 -1.128  0.299  1.822  6.053
#>
#> Coefficients:
#>             Estimate Std. Error t value Pr(>|t|)
#> (Intercept) 13.7253    8.2176  1.670 0.113184
#> True_score.Pre 0.7578    0.1801  4.208 0.000591 ***
#> GroupTreatment 4.6334    1.7272  2.683 0.015736 *
#> ---
#> Signif. codes:
```

16.6. WHAT GOES INSIDE THE MEASUREMENT ERROR (OR CONTROL GROUP CHANGE OR RESIDUALS)

```
#> 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 '' 1  
#>  
#> Residual standard error: 3.764 on 17 degrees of freedom  
#> Multiple R-squared:  0.6497, Adjusted R-squared:  0.6085  
#> F-statistic: 15.76 on 2 and 17 DF, p-value: 0.0001342
```

The model with measured scores is the model we have used so far (`model4`), but I am repeating it here for the sake of easier comprehension:

```
model_measured <- lm(Measured_score.Post ~ Measured_score.Pre +  
  Group, RCT_data)  
  
summary(model_measured)  
#>  
#> Call:  
#> lm(formula = Measured_score.Post ~ Measured_score.Pre + Group,  
#>   data = RCT_data)  
#>  
#> Residuals:  
#>    Min      1Q  Median      3Q     Max  
#> -6.5827 -3.5413  0.3856  2.3547  7.9219  
#>  
#> Coefficients:  
#>             Estimate Std. Error t value Pr(>|t|)  
#> (Intercept) 13.1094    8.5876  1.527 0.145262  
#> Measured_score.Pre 0.7774    0.1902  4.088 0.000767  
#> GroupTreatment 4.9265    1.9679  2.503 0.022784  
#>  
#> (Intercept)  
#> Measured_score.Pre ***  
#> GroupTreatment *  
#> ---  
#> Signif. codes:  
#> 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 '' 1  
#>  
#> Residual standard error: 4.294 on 17 degrees of freedom  
#> Multiple R-squared:  0.6289, Adjusted R-squared:  0.5853  
#> F-statistic: 14.41 on 2 and 17 DF, p-value: 0.000219
```

Estimate random treatment effect for each model are the following, we can use residuals SD for each group:

```
SD_summary_true <- RCT_data %>% mutate(.resid = residuals(model_true)) %>%  
  group_by(Group) %>% summarise(`Residual SD` = sd(.resid))
```

```
SD_summary_true
#> # A tibble: 2 x 2
#>   Group      `Residual SD`
#>   <fct>     <dbl>
#> 1 Control    1.47
#> 2 Treatment  4.96
```

The SD of the Control group residuals using the True scores should only involve random non-treatment effect, which is in our case equal to 1cm, and SD of the Treatment group residuals should be 5cm.

Let's see what's the case with the model that uses Measured scores:

```
SD_summary_measured <- RCT_data %>% mutate(.resid = residuals(model_measured)) %>%
  group_by(Group) %>% summarise(`Residual SD` = sd(.resid))

SD_summary_measured
#> # A tibble: 2 x 2
#>   Group      `Residual SD`
#>   <fct>     <dbl>
#> 1 Control    2.41
#> 2 Treatment  5.39
```

The SD of the Control group residuals using the measured scores should now involve random non-treatment effect (1cm), 2 x biological variation (1.5cm) and 2 x instrumentation noise (0.5cm), which is around $\sqrt{1^2 + 2 * 1.5^2 + 2 * 0.5^2}$, or 2.45cm.

The SD of the Treatment group residuals using the measured scores should now involve random non-treatment effect (1cm), 2 x biological variation (1.5cm), 2 x instrumentation noise (0.5cm), and additional random treatment effect (5cm) which is around $\sqrt{1^2 + 2 * 1.5^2 + 2 * 0.5^2 + 5^2}$, or 5.57cm.

Estimated random treatment effects:

```
sqrt(SD_summary_true$`Residual SD`[2]^2 - SD_summary_true$`Residual SD`[1]^2)
#> [1] 4.738641
```

```
sqrt(SD_summary_measured$`Residual SD`[2]^2 - SD_summary_measured$`Residual SD`[1]^2)
#> [1] 4.816519
```

As can be seen, these are exactly the same. So, using measures scores and SD of control group change or residuals can *recover* true random treatment effect value.

16.7 Prediction perspective 2

Let's see what happens when we run this RCT through predictive model.

```
model5 <- cv_model(Measured_score.Post ~ Measured_score.Pre +
  Group, RCT_data, SESOI_lower = -5, SESOI_upper = 5, control = model_control(seed = 1667,
  cv_folds = 3, cv_repeats = 10, cv_strata = factor(RCT_data$Group)))

prediction_RCT <- RCT_predict(model5, new_data = RCT_data,
  outcome = "Measured_score.Post", group = "Group", treatment_label = "Treatment",
  control_label = "Control", subject_label = RCT_data$Athlete)

prediction_RCT
#> Training data consists of 3 predictors and 20 observations. Cross-Validation of the model was
#>
#> Model performance:
#>
#>           metric      training training.pooled
#>             MBE -9.947541e-15   3.055367e-15
#>             MAE  3.175698e+00   3.037705e+00
#>             RMSE 3.959134e+00   3.786361e+00
#>             PPER  7.666283e-01   8.120334e-01
#> SESOI to RMSE 2.525805e+00   2.641058e+00
#>             R-squared 6.289308e-01   6.606104e-01
#>             MinErr -7.921899e+00   -1.032365e+01
#>             MaxErr  6.582672e+00   8.806128e+00
#>             MaxAbsErr 7.921899e+00   1.032365e+01
#> testing.pooled          mean            SD            min
#> -0.1404122 -0.02732047 2.12193361 -3.5145897
#>  3.8691755  3.90512133 0.93683875  2.6186933
#>  4.7871794  4.73835660 0.85876892  3.4284211
#>  0.7012521  0.63888951 0.07928795  0.4413183
#>  2.0889127  2.17362777 0.37022957  1.3710286
#>  0.4579491  0.21880124 0.64551950 -1.5780914
#> -10.0942651 -6.76317807 2.33015461 -10.0942651
#>  9.2485263  6.05923474 2.11053360  0.1401500
#> 10.0942651  8.02937811 1.17919840  5.5787979
#>             max
#>  3.3371789
#>  6.8884587
#>  7.2937938
#>  0.7862394
#>  2.9167945
#>  0.7986431
#> -2.7453015
```

```

#> 9.2485263
#> 10.0942651
#>
#> Individual model results:
#>
#>   subject      group observed predicted residual
#> Athlete 01 Treatment 47.54084 54.12352 6.5826720
#> Athlete 02 Control 50.47892 47.33454 -3.1443811
#> Athlete 03 Treatment 58.19673 57.64224 -0.5544865
#> Athlete 04 Control 39.83268 40.93613 1.1034457
#> Athlete 05 Treatment 52.43478 56.13132 3.6965427
#> Athlete 06 Control 45.34422 45.12752 -0.2166979
#> Athlete 07 Treatment 53.97310 48.13585 -5.8372501
#> Athlete 08 Control 48.35425 47.34404 -1.0102111
#> Athlete 09 Treatment 65.05732 58.93766 -6.1196617
#> Athlete 10 Control 46.69332 50.18284 3.4895230
#> Athlete 11 Treatment 57.82698 58.90031 1.0733333
#> Athlete 12 Control 49.16140 48.36452 -0.7968803
#> Athlete 13 Treatment 46.24005 50.92561 4.6855567
#> Athlete 14 Control 57.44395 54.26840 -3.1755549
#> Athlete 15 Treatment 50.02857 56.51530 6.4867253
#> Athlete 16 Control 37.56101 41.57384 4.0128333
#> Athlete 17 Treatment 59.95868 52.03678 -7.9218991
#> Athlete 18 Control 49.31810 48.42968 -0.8884198
#> Athlete 19 Treatment 53.26809 51.17656 -2.0915325
#> Athlete 20 Control 53.52199 54.14833 0.6263431
#> magnitude counterfactual pITE pITE_magnitude
#> Higher 49.19697 -4.926546 Equivalent
#> Equivalent 52.26108 4.926546 Equivalent
#> Equivalent 52.71569 -4.926546 Equivalent
#> Equivalent 45.86267 4.926546 Equivalent
#> Equivalent 51.20477 -4.926546 Equivalent
#> Equivalent 50.05407 4.926546 Equivalent
#> Lower 43.20931 -4.926546 Equivalent
#> Equivalent 52.27059 4.926546 Equivalent
#> Lower 54.01111 -4.926546 Equivalent
#> Equivalent 55.10938 4.926546 Equivalent
#> Equivalent 53.97377 -4.926546 Equivalent
#> Equivalent 53.29107 4.926546 Equivalent
#> Equivalent 45.99907 -4.926546 Equivalent
#> Equivalent 59.19494 4.926546 Equivalent
#> Higher 51.58875 -4.926546 Equivalent
#> Equivalent 46.50039 4.926546 Equivalent
#> Lower 47.11023 -4.926546 Equivalent
#> Equivalent 53.35623 4.926546 Equivalent

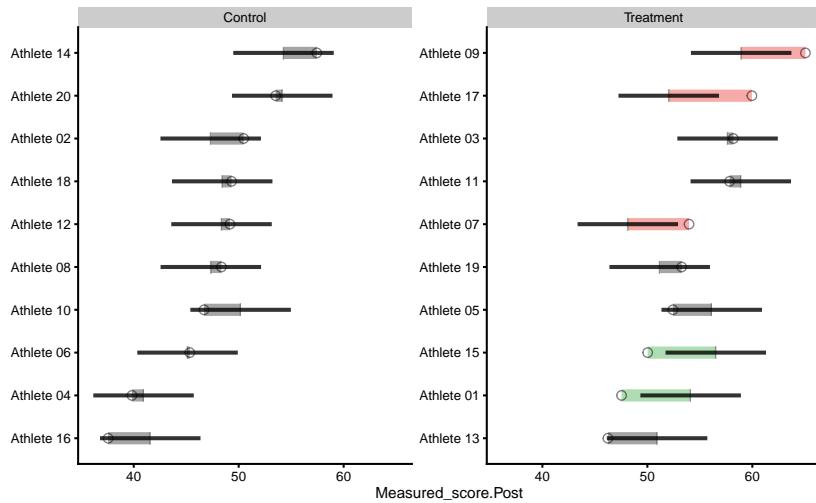
```

```
#>   Equivalent      46.25001 -4.926546   Equivalent
#>   Equivalent      59.07488  4.926546   Equivalent
#>
#> #> Summary of residuals per RCT group:
#>
#>   group      mean      SD
#>   Control -9.237121e-15 2.411836
#>   Treatment -1.065814e-14 5.386633
#>
#> #> Summary of counterfactual effects of RCT group:
#>
#>   group      pATE pVTE
#>   Treatment -4.926546    0
#>   Control   4.926546    0
#>   pooled     4.926546    0
#>
#> #> Treatment effect summary
#>
#> #> Average Treatment effect: 4.926546
#> #> Variable Treatment effect: 0
#> #> Random Treatment effect: 4.816519
```

The prediction performance in this case is much worse (compared to previous RCT example), since now there are systematic and random treatment and non-treatment effects. Although we can *explain* the RCT components, we cannot use it to predict individual responses.

Let's check the individual predictions:

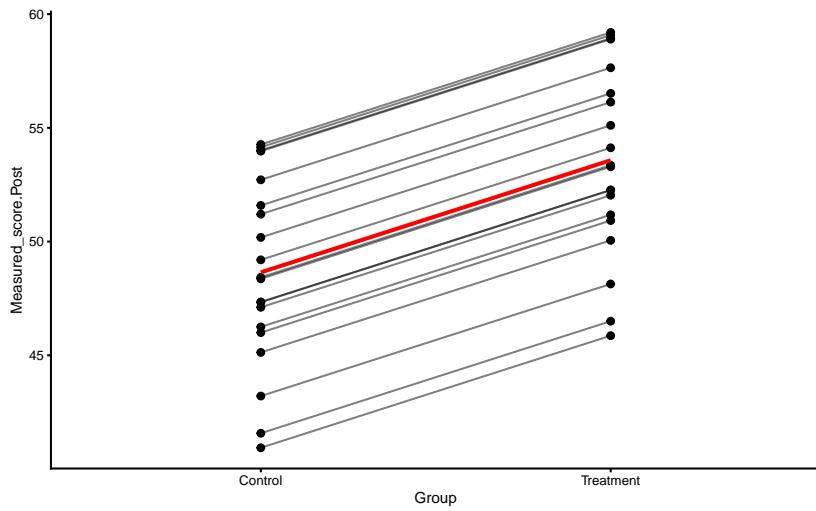
```
plot(prediction_RCT, "prediction")
```



We can now see that certain residuals are larger than SESOI (indicated by green or red color on the figure).

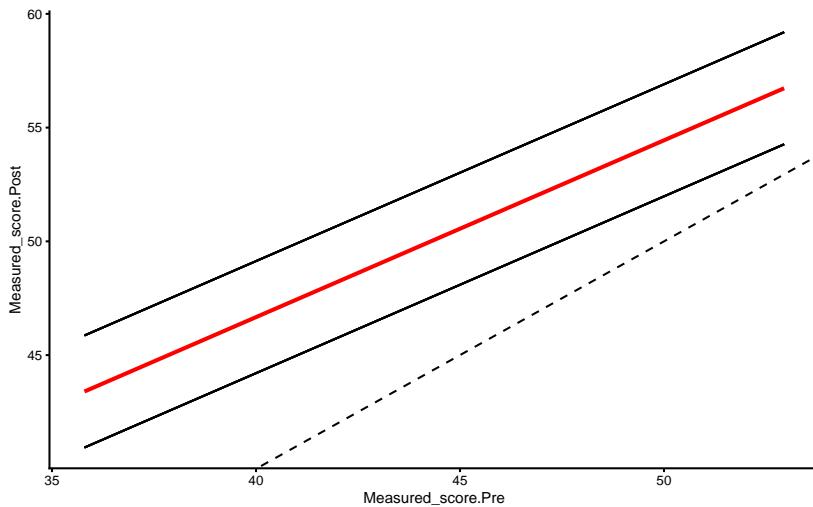
Here is the PDP+ICE plot:

```
plot(prediction_RCT, "pdp+ice")
```



Parallel lines indicate that we predict that each individual will have same treatment effect (indicated by pVTE as well as with the arrows of same length in the counterfactual plot that follows). If we plot PDP+ICE for the measured Pre-test, we will get the following figure:

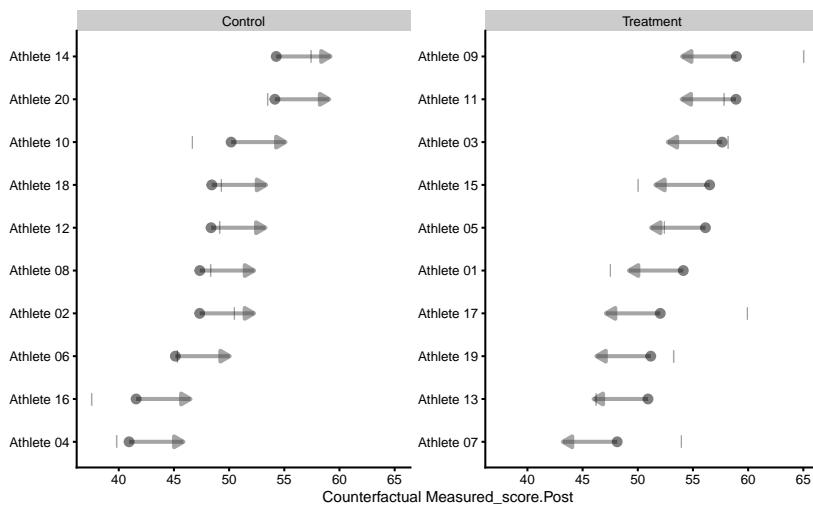
```
plot(prediction_RCT, "pdp+ice", predictor = "Measured_score.Pre") +
  geom_abline(slope = 1, linetype = "dashed")
```



Two thin lines indicate two groups and the gap between them represents the systematic treatment effect.

Estimating individual counterfactual effects when switching group:

```
plot(prediction_RCT, "counterfactual")
```



As can be seen, counterfactual plot depicts same treatment effect for every individual.

16.8 Making it more complex by adding covariate

To make this more complex, let's assume that besides randomizing athletes to Treatment and Control group, we have also measured their lower body strength using relative back squat 1RM. This variable represents our covariate, and we can expect (i.e. we hypothesize) that individuals with higher relative strength will manifest bigger improvements from the plyometric intervention.

The following code represented DGP, where the effects of the training intervention depend on the 1RM squat (i.e. the higher the relative squat, the higher the training intervention). But on top of that, we are going to reduce the improvement based on the initial jump height (i.e. the higher someone jumps, the less improvement over time). It is assumed that squat 1RM and Pre-training jump height are unrelated (independent). Random components of the treatment and non-treatment effects are zero, and the only variance in the measured scores is due to biological noise and instrumentation error.

```
set.seed(16)

n_subjects <- 20

# These stay the same
instrumentation_noise <- 0.5
biological_variation <- 1.5

# -----
# These are not used in this DGP
# Treatment effect
treatment_systematic <- 0
treatment_random <- 0

# Non-treatment effect
non_treatment_systematic <- 0
non_treatment_random <- 0
#-----

RCT_data <- tibble(
  Athlete = paste(
    "Athlete",
    str_pad(
      string = seq(1, n_subjects),
      width = 2,
      pad = "0"
    )
  )
)
```

```

),
Group = rep(c("Treatment", "Control"), length.out = n_subjects),

# Strength covariate
Squat_1RM_relative = rnorm(n_subjects, 1, 0.4),
# True score
True_score.Pre = rnorm(n_subjects, 45, 5),

# Treatment effect
Treatment_effect = (10 - 0.002 * True_score.Pre^2 + 1) * Squat_1RM_relative,
Non_treatment_effect = 15 - 0.005 * True_score.Pre^2,

# Calculate the change in true score
True_score.Change = if_else(
  Group == "Treatment",
  # Treatment group is a sum of treatment and non-treatment effects
  Treatment_effect + Non_treatment_effect,
  # While control group only get non-treatment effects
  Non_treatment_effect
),

True_score.Post = True_score.Pre + True_score.Change,

# Manifested score
Manifested_score.Pre = True_score.Pre + rnorm(n_subjects, 0, biological_variation),
Manifested_score.Post = True_score.Post + rnorm(n_subjects, 0, biological_variation),
Manifested_score.Change = Manifested_score.Post - Manifested_score.Pre,

# Measured score
Measured_score.Pre = Manifested_score.Pre + rnorm(n_subjects, 0, instrumentation_noise),
Measured_score.Post = Manifested_score.Post + rnorm(n_subjects, 0, instrumentation_noise),
Measured_score.Change = Measured_score.Post - Measured_score.Pre
)

RCT_data$Group <- factor(RCT_data$Group)

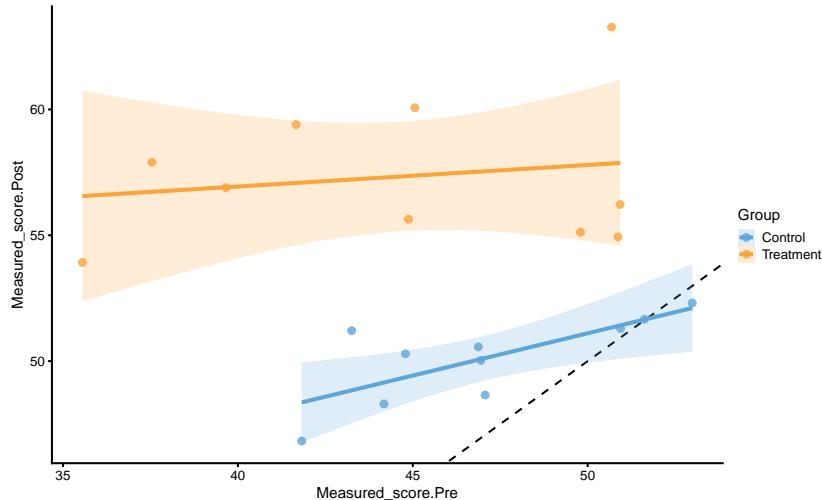
head(RCT_data)
#> # A tibble: 6 x 14
#>   Athlete Group Squat_1RM_relat~ True_score.Pre
#>   <chr>   <fct>          <dbl>        <dbl>
#> 1 Athlet~ Trea~      1.19       36.8
#> 2 Athlet~ Cont~     0.950      43.4
#> 3 Athlet~ Trea~      1.44       44.1
#> 4 Athlet~ Cont~     0.422      52.4
#> 5 Athlet~ Trea~      1.46       40.7

```

```
#> 6 Athlet~ Cont~          0.813      52.6
#> # ... with 10 more variables: Treatment_effect <dbl>,
#> #   Non_treatment_effect <dbl>,
#> #   True_score.Change <dbl>, True_score.Post <dbl>,
#> #   Manifested_score.Pre <dbl>,
#> #   Manifested_score.Post <dbl>,
#> #   Manifested_score.Change <dbl>,
#> #   Measured_score.Pre <dbl>,
#> #   Measured_score.Post <dbl>,
#> #   Measured_score.Change <dbl>
```

Let's plot before we jump into the analysis. In the next plot we will depict measured Post-test (y-axis) and measured Pre-test (x-axis) per group.

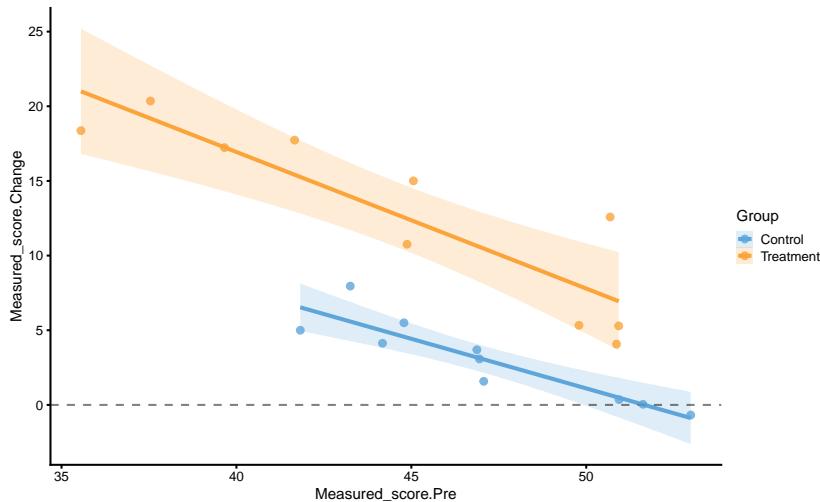
```
ggplot(RCT_data, aes(x = Measured_score.Pre, y = Measured_score.Post,
color = Group, fill = Group)) + theme_cowplot(8) + geom_abline(slope = 1,
linetype = "dashed") + geom_smooth(method = "lm", se = TRUE,
alpha = 0.2) + geom_point(alpha = 0.8) + scale_color_manual(values = c(Treatment =
Control = "#5DA5DA")) + scale_fill_manual(values = c(Treatment = "#FAA43A",
Control = "#5DA5DA"))
```



Might be also usable to plot the change score:

```
ggplot(RCT_data, aes(x = Measured_score.Pre, y = Measured_score.Change,
color = Group, fill = Group)) + theme_cowplot(8) + geom_smooth(method = "lm",
se = TRUE, alpha = 0.2) + geom_hline(yintercept = 0,
linetype = "dashed", alpha = 0.5) + geom_point(alpha = 0.8) +
```

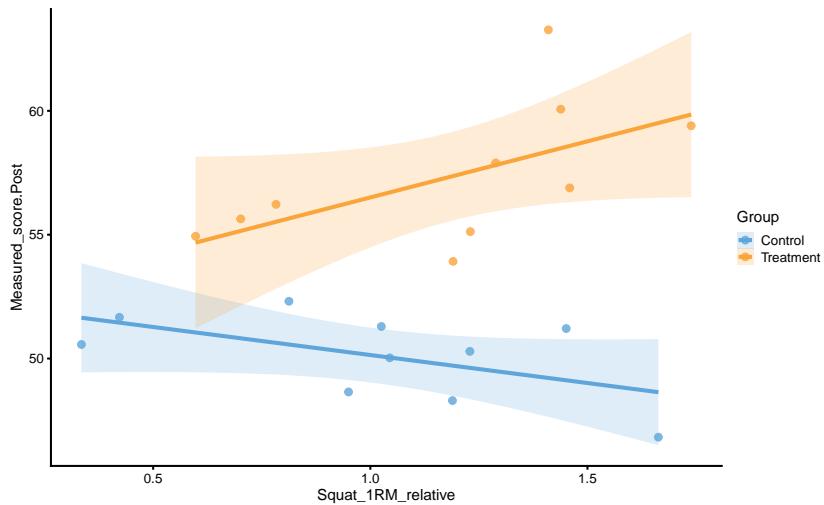
```
scale_color_manual(values = c(Treatment = "#FAA43A",
                             Control = "#5DA5DA")) + scale_fill_manual(values = c(Treatment = "#FAA43A",
                             Control = "#5DA5DA"))
```



From these two graph we can see that as one has higher Pre-test, change scores gets smaller (i.e. effect decreases).

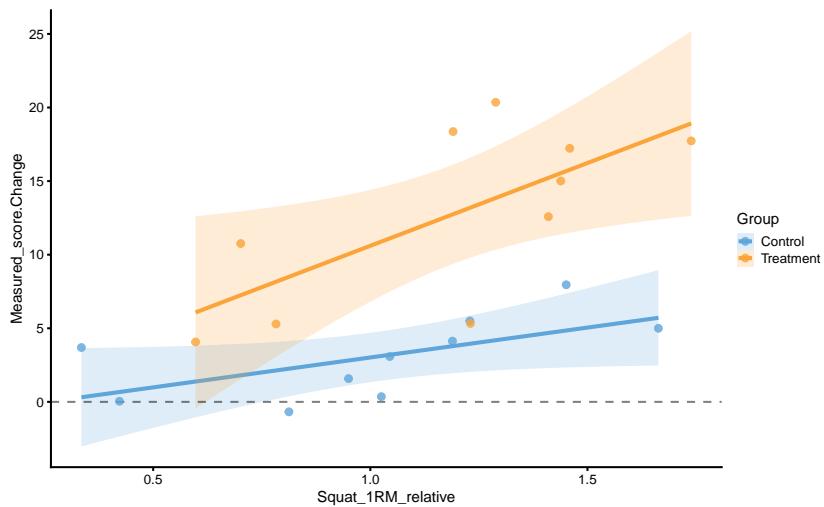
Let's plot the 1RM strength relationship to Post-test:

```
ggplot(RCT_data, aes(x = Squat_1RM_relative, y = Measured_score.Post,
                      color = Group, fill = Group)) + theme_cowplot(8) + geom_smooth(method = "lm",
                      se = TRUE, alpha = 0.2) + geom_point(alpha = 0.8) + scale_color_manual(values = c(Treatment =
                      "#FAA43A", Control = "#5DA5DA")) + scale_fill_manual(values = c(Treatment = "#FAA43A",
                      Control = "#5DA5DA"))
```



As can be seen from the figure, there is interaction between 1RM and group (the lines are not parallel). Let's check with the change score:

```
ggplot(RCT_data, aes(x = Squat_1RM_relative, y = Measured_score.Change,
color = Group, fill = Group)) + theme_cowplot(8) + geom_smooth(method = "lm",
se = TRUE, alpha = 0.2) + geom_hline(yintercept = 0,
linetype = "dashed", alpha = 0.5) + geom_point(alpha = 0.8) +
scale_color_manual(values = c(Treatment = "#FAA43A",
Control = "#5DA5DA")) + scale_fill_manual(values = c(Treatment = "#FAA43A",
Control = "#5DA5DA"))
```



With change score we have *controlled* for Pre-test, which gives us information that the stronger someone is, the higher the improvement. This is even more evident for the Treatment group.

Let's do the usual analysis and check the effects. Let's first perform the method of the differences (which uses change score `mean` and SD):

```
simple_RCT <- bmbstats::RCT_analysis(data = RCT_data, group = "Group",
  treatment_label = "Treatment", control_label = "Control",
  pre_test = "Measured_score.Pre", post_test = "Measured_score.Post",
  SESOI_lower = -5, SESOI_upper = 5, estimator_function = bmbstats::RCT_estimators_simple,
  control = model_control(seed = 1667))
#> [1] "All values of t are equal to 5 \n Cannot calculate confidence intervals"
#> [1] "All values of t are equal to 10 \n Cannot calculate confidence intervals"

simple_RCT
#> Bootstrap with 2000 resamples and 95% bca confidence intervals.
#>
#>           estimator      value     lower
#> SESOI lower -5.000000000          NA
#> SESOI upper  5.000000000          NA
#> SESOI range   10.000000000         NA
#> Systematic effect  9.60660366 5.36900820
#>           Random effect  5.39012073 3.51072089
#> Systematic effect to SESOI  0.96066037 0.53690082
#> SESOI to Random effect  1.85524601 1.43972066
#>           pLower  0.01200102 0.00037884
#>           pEquivalent  0.19545959 0.02005083
#>           pHIGHER  0.79253939 0.46809992
#>           upper
#>           NA
#>           NA
#>           NA
#> 13.44947544
#> 6.92558325
#> 1.34494754
#> 3.06650272
#> 0.05838599
#> 0.46036640
#> 0.97792031
```

From our DGP we know that there is no random treatment effect, but only systematic treatment effect depending on the Pre-test and relative strength levels. But with this analysis we do not know that and we use only `mean` change to estimate treatment effects, and all the residuals around the `mean` (i.e. total

variance) are assumed to be random error. This is *unexplained* variance (by this model). But we do know we can explain this variance with better model specification.

If we use linear regression model (using `bmbstats::RCT_estimators_lm`), we will address the effect of the Pre-test on the Post-test. But before, let's just show model coefficients:

```
model6 <- lm(Measured_score.Post ~ Measured_score.Pre + Group,
               RCT_data)

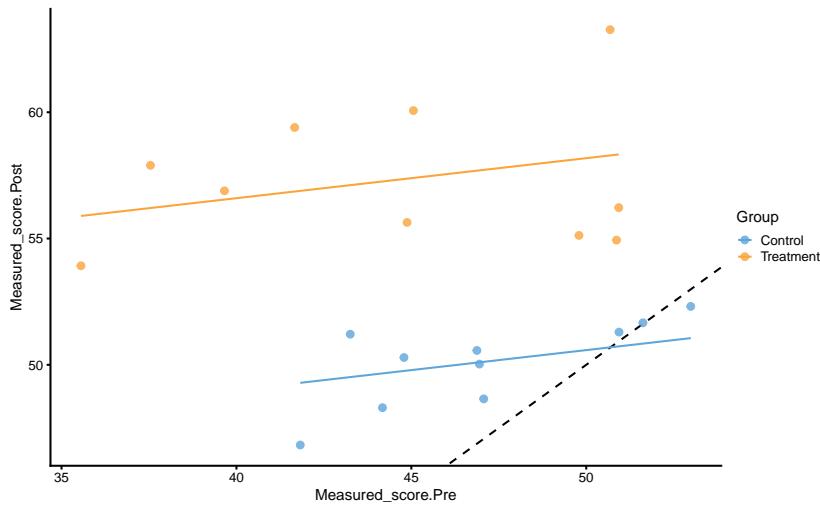
summary(model6)
#>
#> Call:
#> lm(formula = Measured_score.Post ~ Measured_score.Pre + Group,
#>      data = RCT_data)
#>
#> Residuals:
#>    Min     1Q   Median     3Q    Max
#> -3.3819 -1.7924  0.4125  1.3648  4.9797
#>
#> Coefficients:
#>             Estimate Std. Error t value Pr(>|t|)
#> (Intercept) 42.6606   5.2220  8.169 2.74e-07
#> Measured_score.Pre 0.1584   0.1099  1.441  0.168
#> GroupTreatment 7.5994   1.0578  7.184 1.53e-06
#>
#> (Intercept) ***
#> Measured_score.Pre
#> GroupTreatment ***
#> ---
#> Signif. codes:
#> 0 '****' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 2.291 on 17 degrees of freedom
#> Multiple R-squared:  0.7527, Adjusted R-squared:  0.7236
#> F-statistic: 25.87 on 2 and 17 DF,  p-value: 6.963e-06
```

Visually, this model looks like this:

```
RCT_data$.pred <- predict(model6, newdata = RCT_data)

ggplot(RCT_data, aes(x = Measured_score.Pre, y = Measured_score.Post,
                      color = Group, fill = Group)) + theme_cowplot(8) + geom_abline(slope = 1,
                      linetype = "dashed") + geom_point(alpha = 0.8) + geom_line(aes(y = .pred)) +
                      scale_color_manual(values = c(Treatment = "#FAA43A",
```

```
Control = "#5DA5DA")) + scale_fill_manual(values = c(Treatment = "#FAA43A",
Control = "#5DA5DA"))
```



Estimated random treatment effect is equal to:

```
SD_summary <- RCT_data %>% mutate(.resid = residuals(model6)) %>%
  group_by(Group) %>% summarise(`Residual SD` = sd(.resid))

sqrt(SD_summary$`Residual SD`[2]^2 - SD_summary$`Residual SD`[1]^2)
#> [1] 2.521028
```

Now, let's run this model using the `bmbstats::RCT_estimators_lm` and `bmbstats::RCT_analysis`:

```
regression_RCT <- bmbstats::RCT_analysis(data = RCT_data,
  group = "Group", treatment_label = "Treatment", control_label = "Control",
  pre_test = "Measured_score.Pre", post_test = "Measured_score.Post",
  SESOI_lower = -5, SESOI_upper = 5, estimator_function = bmbstats::RCT_estimators_lm,
  control = model_control(seed = 1667))
#> [1] "All values of t are equal to 5 \n Cannot calculate confidence intervals"
#> [1] "All values of t are equal to 10 \n Cannot calculate confidence intervals"

regression_RCT
#> Bootstrap with 2000 resamples and 95% bca confidence intervals.
#>
```

```

#>           estimator      value      lower
#> SESOI lower -5.000000e+00      NA
#> SESOI upper  5.000000e+00      NA
#> SESOI range   1.000000e+01      NA
#>     Systematic effect  7.599370e+00  5.559460e+00
#>     Random effect    2.521028e+00  1.281688e+00
#> Systematic effect to SESOI 7.599370e-01  5.559460e-01
#> SESOI to Random effect 3.966636e+00 -2.288091e+01
#>           pLower 3.995354e-05  7.377399e-10
#>           pEquivalent 1.576864e-01  2.068126e-03
#>           pHiger  8.422736e-01  3.994218e-05
#>           upper
#>           NA
#>           NA
#>           NA
#> 10.4054050
#> 3.7842396
#> 1.0405405
#> 9.9649843
#> 1.0000000
#> 0.4686692
#> 0.9902346

```

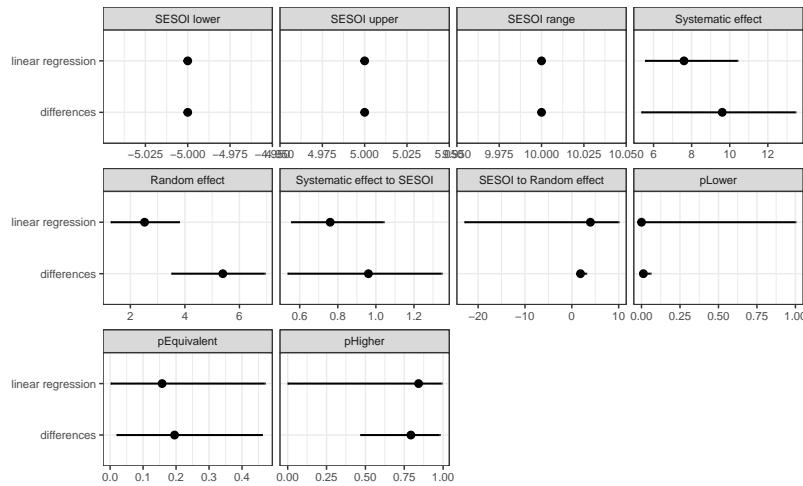
Let's plot the estimated from the method of differences and linear regression model for easier comparison:

```

compare_methods <- rbind(data.frame(method = "differences",
  simple_RCT$estimators), data.frame(method = "linear regression",
  regression_RCT$estimators))

ggplot(compare_methods, aes(y = method, x = value)) + theme_bw(8) +
  geom_errorbarh(aes(xmax = upper, xmin = lower), color = "black",
  height = 0) + geom_point() + xlab("") + ylab("") +
  facet_wrap(~estimator, scales = "free_x")

```



Please note that the estimated systematic treatment effect is smaller for the linear regression method compared to method of differences. It is the opposite for the random treatment effect estimate. This is because linear regression method estimates the effect of the group while controlling for the Pre-test.

Let's now add relative 1RM strength to the mix:

```
model17 <- lm(Measured_score.Post ~ Measured_score.Pre + Squat_1RM_relative +
  Group, RCT_data)

summary(model17)
#>
#> Call:
#> lm(formula = Measured_score.Post ~ Measured_score.Pre + Squat_1RM_relative +
#>     Group, data = RCT_data)
#>
#> Residuals:
#>    Min      1Q  Median      3Q     Max
#> -3.6732 -1.3602  0.1547  1.5569  3.7951
#>
#> Coefficients:
#>             Estimate Std. Error t value Pr(>|t|)
#> (Intercept) 35.2743    6.7717   5.209 8.61e-05
#> Measured_score.Pre 0.2621    0.1231   2.128  0.0492
#> Squat_1RM_relative 2.4779    1.5351   1.614  0.1260
#> GroupTreatment  7.4215    1.0171   7.297 1.79e-06
#>
#> (Intercept) ***
#> Measured_score.Pre *
```

```
#> Squat_1RM_relative
#> GroupTreatment      ***
#> ---
#> Signif. codes:
#> 0 '****' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 2.19 on 16 degrees of freedom
#> Multiple R-squared:  0.7873, Adjusted R-squared:  0.7474
#> F-statistic: 19.74 on 3 and 16 DF,  p-value: 1.259e-05
```

Please notice the slightly improved RSE of this model (indicating better model fit). Let's check the estimate random treatment effect:

```
SD_summary <- RCT_data %>% mutate(.resid = residuals(model7)) %>%
  group_by(Group) %>% summarise(`Residual SD` = sd(.resid))

sqrt(SD_summary$`Residual SD`[2]^2 - SD_summary$`Residual SD`[1]^2)
#> [1] 1.59975
```

This implies that with this model we were able to explain more of the treatment effect. But we do know there is interaction between Group and relative 1RM. We can model that:

```
model8 <- lm(
  Measured_score.Post ~ Measured_score.Pre + Squat_1RM_relative + Squat_1RM_relative:Group
  # OR we can specify the model with this
  # Measured_score.Post ~ Measured_score.Pre + Squat_1RM_relative * Group
  RCT_data
)

summary(model8)
#>
#> Call:
#> lm(formula = Measured_score.Post ~ Measured_score.Pre + Squat_1RM_relative +
#>     Squat_1RM_relative:Group + Group, data = RCT_data)
#>
#> Residuals:
#>    Min      1Q  Median      3Q     Max
#> -3.8832 -0.8725 -0.0257  0.9302  2.8875
#>
#> Coefficients:
#>              Estimate Std. Error
#> (Intercept) 38.07464   5.35343
```

```
#> Measured_score.Pre          0.26809   0.09618
#> Squat_1RM_relative        -0.56615   1.50382
#> GroupTreatment            -0.20765   2.41029
#> Squat_1RM_relative:GroupTreatment 6.89774   2.05750
#>                               t value Pr(>/t|)
#> (Intercept)                7.112 3.55e-06 ***
#> Measured_score.Pre          2.787  0.01381 *
#> Squat_1RM_relative         -0.376  0.71183
#> GroupTreatment             -0.086  0.93248
#> Squat_1RM_relative:GroupTreatment 3.352  0.00436 **
#> ---
#> Signif. codes:
#> 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 1.71 on 15 degrees of freedom
#> Multiple R-squared:  0.8784, Adjusted R-squared:  0.846
#> F-statistic: 27.09 on 4 and 15 DF, p-value: 1.039e-06
```

RSE of this model improved even further. Let's calculate the random treatment effect:

```
SD_summary <- RCT_data %>% mutate(.resid = residuals(model8)) %>%
  group_by(Group) %>% summarise(`Residual SD` = sd(.resid))

sqrt(SD_summary$`Residual SD`[2]^2 - SD_summary$`Residual SD`[1]^2)
#> [1] 1.476142
```

We managed to explain mode of the treatment effect (i.e. reduce the random component), but how do we not estimate the systematic effect? Now we have interaction term, and we introduced *indirect* effect of the Group:1RM, so we cannot simply quantify it with a single number.

But maybe the predictive perspective can help?

16.9 Prediction perspective 3

Before analyzing the most complex model specification, let's build from the ground-up again using *baseline* model first. This model predicts Post-test to be the same for everyone using Group as predictor. This model is useful since it gives us the worst prediction we can use to judge the other models.

Since we are going to perform same modeling using different specification, to avoid repeating the code (which I have done on purpose thorough this book, but will avoid doing here), I will write down the function:

```

model_RCT <- function(formula) {
  model <- cv_model(formula, RCT_data, SESOI_lower = -5,
    SESOI_upper = 5, control = model_control(seed = 1667,
    cv_folds = 5, cv_repeats = 10, cv_strata = factor(RCT_data$Group)))

  RCT <- RCT_predict(model, new_data = RCT_data, outcome = "Measured_score.Post",
    group = "Group", treatment_label = "Treatment", control_label = "Control",
    subject_label = RCT_data$Athlete)

  return(RCT)
}

```

Here is our baseline model:

```

base_model <- model_RCT(Measured_score.Post ~ Group)

base_model
#> Training data consists of 2 predictors and 20 observations. Cross-Validation of the
#>
#> Model performance:
#>
#>           metric      training training.pooled
#>           MBE -4.618506e-15  1.634262e-15
#>           MAE  1.793742e+00  1.768937e+00
#>           RMSE 2.238021e+00  2.206270e+00
#>           PPER  9.5777526e-01  9.762111e-01
#> SESOI to RMSE 4.468233e+00  4.532536e+00
#> R-squared 7.224490e-01  7.302684e-01
#> MinErr -5.933469e+00 -6.532374e+00
#> MaxErr  3.414524e+00  3.926733e+00
#> MaxAbsErr 5.933469e+00  6.532374e+00
#> testing.pooled          mean            SD       min
#> 1.634259e-15  1.634238e-15  1.33279182 -2.5926409
#> 1.9977720e+00 1.9977720e+00  0.55544631  1.0781811
#> 2.505740e+00 2.393620e+00  0.74868547  1.1928031
#> 9.520871e-01 8.549957e-01  0.08929827  0.6560171
#> 3.990837e+00 4.591613e+00  1.42736849  2.4405997
#> 6.520744e-01 6.595240e-01  0.32999136 -0.5542973
#> -7.016430e+00 -2.794574e+00  2.16480138 -7.0164300
#> 4.141234e+00 2.385040e+00  1.21855651 -0.2376657
#> 7.016430e+00 3.768324e+00  1.58419875  1.4647768
#> max
#> 2.7658984
#> 3.3474423
#> 4.0973536

```



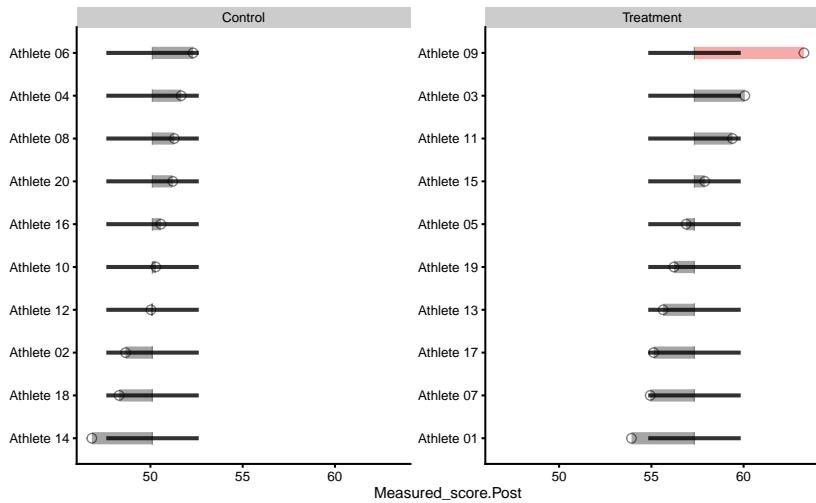
```

#>   Equivalent      50.11469 -7.221484      Lower
#>   Equivalent      57.33617  7.221484      Higher
#>   Equivalent      50.11469 -7.221484      Lower
#>   Equivalent      57.33617  7.221484      Higher
#>   Equivalent      50.11469 -7.221484      Lower
#>   Equivalent      57.33617  7.221484      Higher
#>
#> Summary of residuals per RCT group:
#>
#>   group       mean        SD
#>   Control -2.842214e-15 1.709932
#>   Treatment -6.394885e-15 2.864728
#>
#> Summary of counterfactual effects of RCT group:
#>
#>   group      pATE pVTE
#>   Treatment -7.221484    0
#>   Control    7.221484    0
#>   pooled     7.221484    0
#>
#> Treatment effect summary
#>
#> Average Treatment effect: 7.221484
#> Variable Treatment effect: 0
#> Random Treatment effect: 2.298433

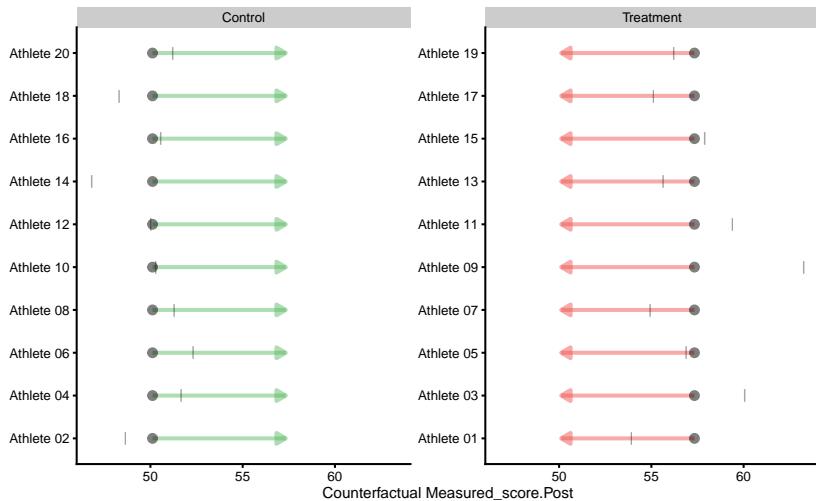
```

Let's plot key model predictions (individual and counterfactual).

```
plot(base_model, "prediction")
```



```
plot(base_model, "counterfactual")
```



As can be seen, base model predict the same Post-test scores for each athlete (depending on the group).

Our next model is the one that uses Pre-test and Group. Let's call it `pre_test_model` (please note that this is the model used in `bmbstats::RCT_estimators_lm` function):

```

pre_test_model <- model_RCT(Measured_score.Post ~ Group +
                               Measured_score.Pre)

pre_test_model
#> Training data consists of 3 predictors and 20 observations. Cross-Validation of the
#>
#> Model performance:
#>
#>      metric      training training.pooled
#>      MBE -1.385555e-14   3.774758e-15
#>      MAE  1.757354e+00   1.700865e+00
#>      RMSE 2.112647e+00   2.066468e+00
#>      PPER  9.675019e-01   9.841744e-01
#> SESOI to RMSE 4.733399e+00   4.839174e+00
#> R-squared 7.526750e-01   7.633690e-01
#> MinErr -4.979659e+00   -5.620519e+00
#> MaxErr  3.381897e+00   4.723784e+00
#> MaxAbsErr 4.979659e+00   5.620519e+00
#> testing.pooled      mean        SD        min
#> 0.03197143 0.03197143 1.25159476 -2.5288076
#> 2.04500207 2.04500207 0.53059190  0.8621495
#> 2.51137986 2.43165066 0.63415044  1.0726499
#> 0.95158915 0.84684948 0.07383527  0.6842390
#> 3.98187473 4.44664689 1.40990552  2.5812789
#> 0.65056316 0.63480891 0.32450934 -0.7834415
#> -6.48925129 -2.73668942 1.95426062 -6.4892513
#> 4.36811806 2.71096150 1.23032364 -0.6916705
#> 6.48925129 3.85285271 1.30595427  1.9061280
#> max
#> 2.8367663
#> 3.2662051
#> 3.8740486
#> 0.9746472
#> 9.3227066
#> 0.9507508
#> 1.9406012
#> 4.3681181
#> 6.4892513
#>
#> Individual model results:
#>
#>      subject group observed predicted residual
#> Athlete 01 Treatment 53.92165 55.89316 1.97150648
#> Athlete 02 Control 48.65099 50.11805 1.46705656
#> Athlete 03 Treatment 60.06640 57.39899 -2.66741213

```

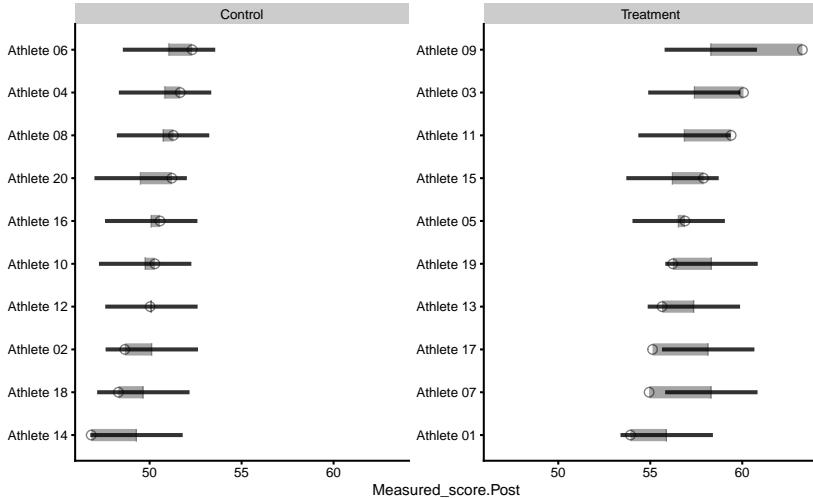
```

#> Athlete 04 Control 51.66451 50.83931 -0.82520345
#> Athlete 05 Treatment 56.88816 56.54332 -0.34484151
#> Athlete 06 Control 52.31237 51.05577 -1.25660375
#> Athlete 07 Treatment 54.93702 58.31892 3.38189720
#> Athlete 08 Control 51.29489 50.73095 -0.56393813
#> Athlete 09 Treatment 63.26964 58.28998 -4.97965920
#> Athlete 10 Control 50.28937 49.75682 -0.53254676
#> Athlete 11 Treatment 59.39720 56.86111 -2.53609024
#> Athlete 12 Control 50.02810 50.09865 0.07054839
#> Athlete 13 Treatment 55.63766 57.37036 1.73269990
#> Athlete 14 Control 46.82518 49.28717 2.46198498
#> Athlete 15 Treatment 57.89749 56.20799 -1.68949873
#> Athlete 16 Control 50.56725 50.08713 -0.48012312
#> Athlete 17 Treatment 55.12290 58.14891 3.02601000
#> Athlete 18 Control 48.30309 49.65954 1.35644850
#> Athlete 19 Treatment 56.22363 58.32902 2.10538825
#> Athlete 20 Control 51.21115 49.51352 -1.69762322
#> magnitude counterfactual pITE pITE_magnitude
#> Equivalent 48.29379 -7.59937 Lower
#> Equivalent 57.71742 7.59937 Higher
#> Equivalent 49.79962 -7.59937 Lower
#> Equivalent 58.43868 7.59937 Higher
#> Equivalent 48.94395 -7.59937 Lower
#> Equivalent 58.65514 7.59937 Higher
#> Equivalent 50.71955 -7.59937 Lower
#> Equivalent 58.33032 7.59937 Higher
#> Equivalent 50.69061 -7.59937 Lower
#> Equivalent 57.35619 7.59937 Higher
#> Equivalent 49.26174 -7.59937 Lower
#> Equivalent 57.69802 7.59937 Higher
#> Equivalent 49.77099 -7.59937 Lower
#> Equivalent 56.88654 7.59937 Higher
#> Equivalent 48.60862 -7.59937 Lower
#> Equivalent 57.68650 7.59937 Higher
#> Equivalent 50.54954 -7.59937 Lower
#> Equivalent 57.25891 7.59937 Higher
#> Equivalent 50.72965 -7.59937 Lower
#> Equivalent 57.11289 7.59937 Higher
#>
#> Summary of residuals per RCT group:
#>
#>      group      mean       SD
#>      Control -1.136866e-14 1.334693
#>      Treatment -1.634257e-14 2.852540
#>

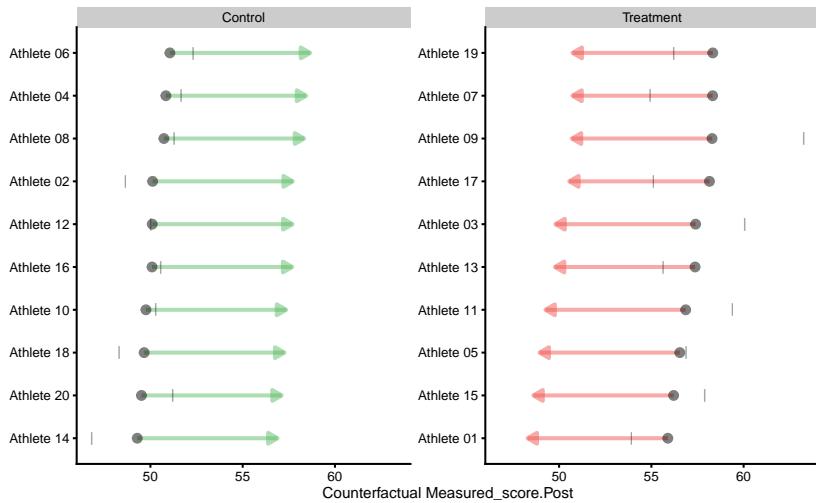
```

```
#> Summary of counterfactual effects of RCT group:
#>
#>      group      pATE      pVTE
#> Treatment -7.59937 5.246187e-15
#> Control    7.59937 2.254722e-15
#> pooled     7.59937 3.992905e-15
#>
#> Treatment effect summary
#>
#> Average Treatment effect: 7.59937
#> Variable Treatment effect: 3.992905e-15
#> Random Treatment effect: 2.521028
```

```
plot(pre_test_model, "prediction")
```

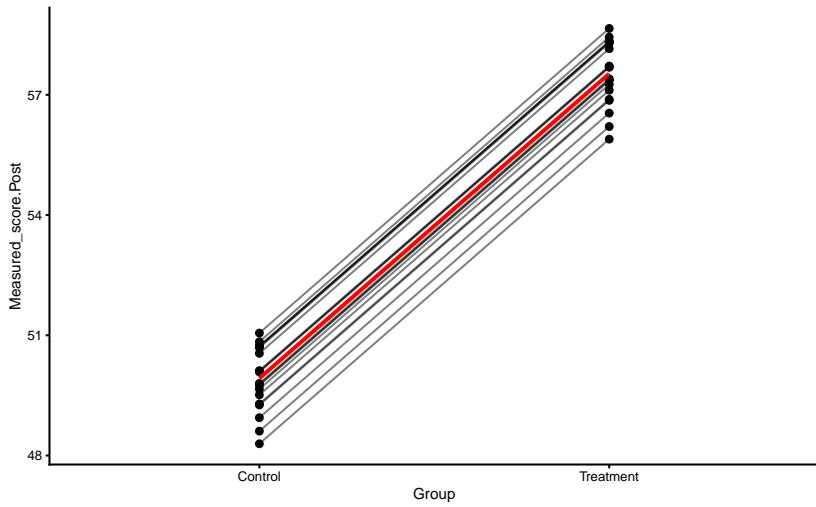


```
plot(pre_test_model, "counterfactual")
```



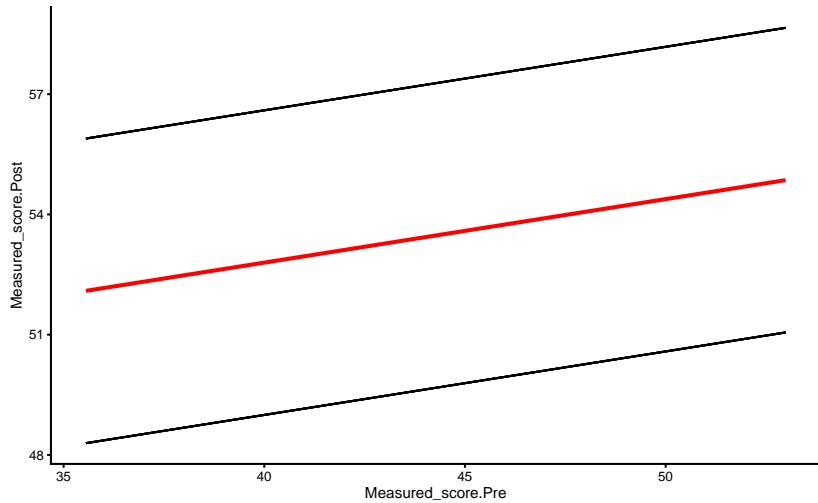
Additional plot we can do is the PDP+ICE for group predictor (i.e. treatment effect):

```
plot(pre_test_model, "pdp+ice")
```



And also for the Pre-test predictor:

```
plot(pre_test_model, "pdp+ice", predictor = "Measured_score.Pre")
```



Next model adds additional predictor (covariate): relative squat 1RM:

```
covariate_model <- model_RCT(Measured_score.Post ~ Group +
  Measured_score.Pre + Squat_1RM_relative)

covariate_model
#> Training data consists of 4 predictors and 20 observations. Cross-Validation of the
#>
#> Model performance:
#>
#>           metric      training training.pooled
#>             MBE -1.492140e-14   3.641497e-15
#>             MAE  1.609713e+00   1.560087e+00
#>             RMSE 1.959142e+00   1.895527e+00
#>             PPER 9.776803e-01   9.914518e-01
#> SESOI to RMSE 5.104275e+00   5.275579e+00
#> R-squared 7.873104e-01   8.008987e-01
#> MinErr -3.795052e+00   -4.671246e+00
#> MaxErr  3.673197e+00   4.356870e+00
#> MaxAbsErr 3.795052e+00   4.671246e+00
#> testing.pooled          mean            SD        min
#> -0.04731864 -0.04731864 1.27184288 -2.4388220
#> 1.99431750  1.99431750 0.67386685  0.3458090
#> 2.51905179  2.40288524 0.76382791  0.4106030
#> 0.95090835  0.84667956 0.08174096  0.6970788
#> 3.96974768  5.02832487 3.49488784  2.6565400
#> 0.64849202  0.58513228 0.56466111 -1.9362660
#> -6.30927941 -2.64672001 1.65641966 -6.3092794
```

```

#>      5.13611940  2.81303001 1.68991662 -0.2922030
#>      6.30927941  3.84840058 1.32366331  0.7157490
#>      max
#>      2.4598851
#>      3.4404583
#>      3.7642949
#>      0.9981811
#>      24.3544266
#>      0.9784156
#>      0.8127197
#>      5.1361194
#>      6.3092794
#>
#> Individual model results:
#>
#>      subject   group observed predicted residual
#> Athlete 01 Treatment 53.92165 54.96555  1.0439001
#> Athlete 02 Control 48.65099 49.96562  1.3146319
#> Athlete 03 Treatment 60.06640 58.07113 -1.9952645
#> Athlete 04 Control 51.66451 49.85167 -1.8128409
#> Athlete 05 Treatment 56.88816 56.70667 -0.1814903
#> Athlete 06 Control 52.31237 51.17699 -1.1353824
#> Athlete 07 Treatment 54.93702 57.50946  2.5724418
#> Athlete 08 Control 51.29489 51.16688 -0.1280040
#> Athlete 09 Treatment 63.26964 59.47459 -3.7950523
#> Athlete 10 Control 50.28937 50.06036 -0.2290056
#> Athlete 11 Treatment 59.39720 57.92560 -1.4715975
#> Athlete 12 Control 50.02810 50.16875  0.1406476
#> Athlete 13 Treatment 55.63766 56.19778  0.5601237
#> Athlete 14 Control 46.82518 50.35886  3.5336766
#> Athlete 15 Treatment 57.89749 55.72955 -2.1679347
#> Athlete 16 Control 50.56725 48.39034 -2.1769094
#> Athlete 17 Treatment 55.12290 58.79610  3.6731974
#> Athlete 18 Control 48.30309 49.79992  1.4968326
#> Athlete 19 Treatment 56.22363 57.98531  1.7616764
#> Athlete 20 Control 51.21115 50.20750 -1.0036463
#>      magnitude counterfactual pITE pITE_magnitude
#> Equivalent      47.54404 -7.421514      Lower
#> Equivalent      57.38714  7.421514      Higher
#> Equivalent      50.64962 -7.421514      Lower
#> Equivalent      57.27318  7.421514      Higher
#> Equivalent      49.28515 -7.421514      Lower
#> Equivalent      58.59851  7.421514      Higher
#> Equivalent      50.08795 -7.421514      Lower
#> Equivalent      58.58840  7.421514      Higher

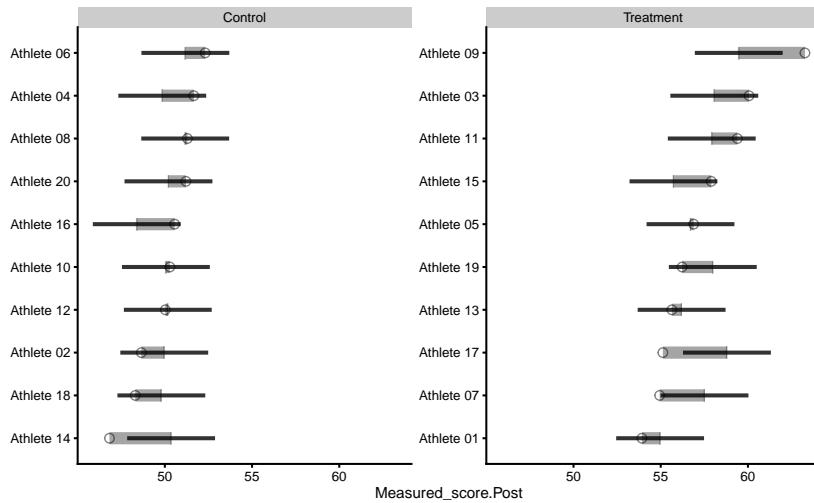
```

```

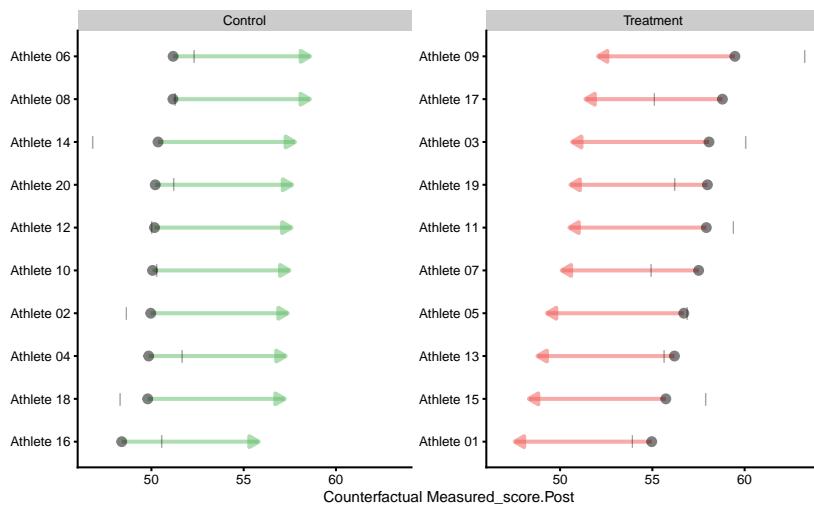
#>   Equivalent      52.05308 -7.421514      Lower
#>   Equivalent      57.48188  7.421514      Higher
#>   Equivalent      50.50408 -7.421514      Lower
#>   Equivalent      57.59026  7.421514      Higher
#>   Equivalent      48.77627 -7.421514      Lower
#>   Equivalent      57.78037  7.421514      Higher
#>   Equivalent      48.30804 -7.421514      Lower
#>   Equivalent      55.81186  7.421514      Higher
#>   Equivalent      51.37458 -7.421514      Lower
#>   Equivalent      57.22144  7.421514      Higher
#>   Equivalent      50.56380 -7.421514      Lower
#>   Equivalent      57.62902  7.421514      Higher
#>
#> Summary of residuals per RCT group:
#>
#>   group       mean        SD
#>   Control -1.492135e-14 1.727747
#>   Treatment -1.492135e-14 2.354636
#>
#> Summary of counterfactual effects of RCT group:
#>
#>   group      pATE      pVTE
#>   Treatment -7.421514 5.617334e-15
#>   Control   7.421514 4.037713e-15
#>   pooled    7.421514 4.890290e-15
#>
#> Treatment effect summary
#>
#> Average Treatment effect: 7.421514
#> Variable Treatment effect: 4.89029e-15
#> Random Treatment effect: 1.59975

```

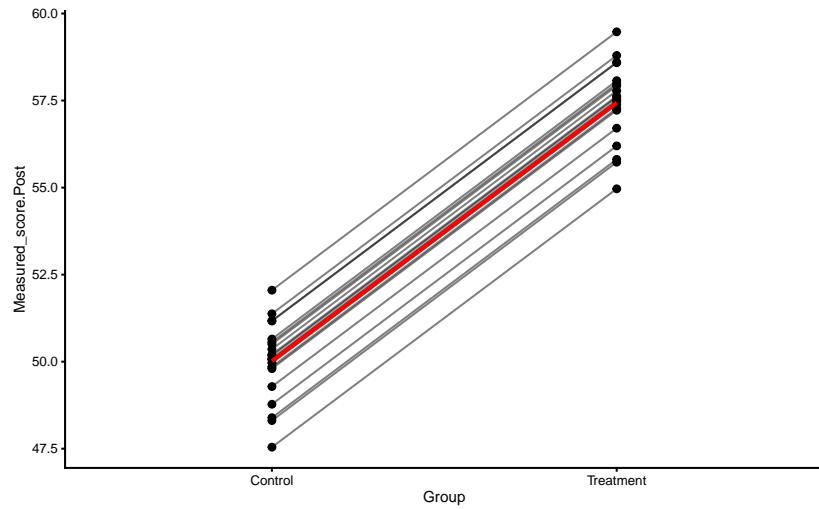
```
plot(covariate_model, "prediction")
```



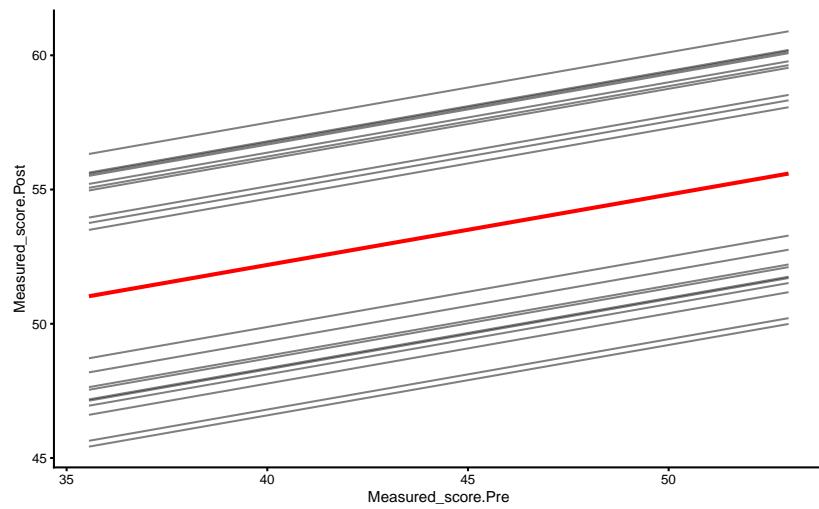
```
plot(covariate_model, "counterfactual")
```



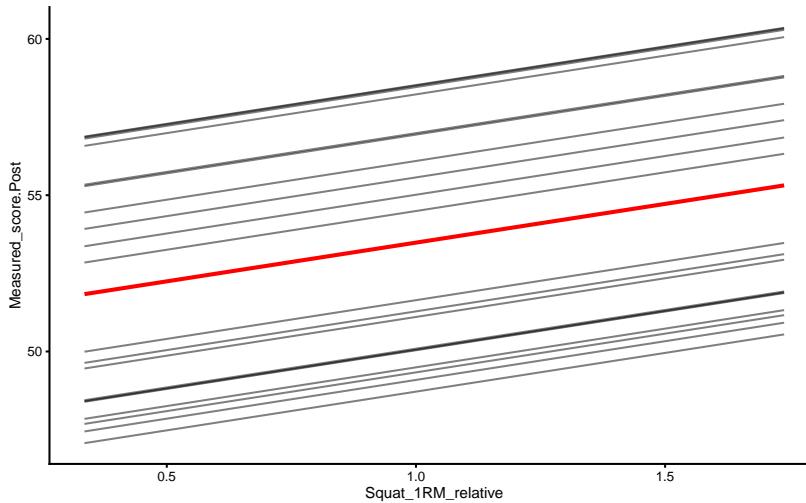
```
plot(covariate_model, "pdp+ice")
```



```
plot(covariate_model, "pdp+ice", predictor = "Measured_score.Pre")
```



```
plot(covariate_model, "pdp+ice", predictor = "Squat_1RM_relative")
```



And the final model is the interaction model:

```
interaction_model <- model_RCT(Measured_score.Post ~ Group *
  Squat_1RM_relative + Measured_score.Pre)

interaction_model
#> Training data consists of 5 predictors and 20 observations. Cross-Validation of the model was
#>
#> Model performance:
#>
#>           metric      training training.pooled
#>           MBE  2.486856e-15  5.107026e-15
#>           MAE  1.102035e+00  1.073541e+00
#>           RMSE 1.481278e+00  1.420252e+00
#>           PPER 9.961488e-01  9.995411e-01
#> SESOI to RMSE 6.750926e+00  7.041003e+00
#> R-squared 8.784129e-01  8.882249e-01
#> MinErr -2.887520e+00 -3.387465e+00
#> MaxErr  3.883197e+00  4.125618e+00
#> MaxAbsErr 3.883197e+00  4.125618e+00
#> testing.pooled      mean          SD        min
#> -0.02897255 -0.02897255 0.96232473 -2.10791777
#> 1.51448758  1.51448758 0.62541112  0.42250638
#> 2.00731144  1.86964221 0.73799039  0.62520553
#> 0.98620762  0.90318747 0.07713261  0.73706941
#> 4.98178798  6.39845655 2.94071918  3.06196160
#> 0.77676962  0.72661171 0.39233243 -1.30891277
#> -4.84781640 -2.22509568 1.45777723 -4.84781640
```

```

#>      5.22505955  2.04804580  1.57273644  0.03320484
#>      5.22505955  3.10011470  1.33675406  1.05228122
#>      max
#>      2.0771169
#>      2.9228564
#>      3.2658803
#>      0.9955572
#>      15.9947402
#>      0.9875223
#>      1.2235186
#>      5.2250596
#>      5.2250596
#>
#> Individual model results:
#>
#>      subject    group observed predicted   residual
#> Athlete 01 Treatment 53.92165  54.93714  1.01549088
#> Athlete 02 Control  48.65099  50.15570  1.50470919
#> Athlete 03 Treatment 60.06640  59.05490 -1.01150268
#> Athlete 04 Control  51.66451  51.67481  0.01030262
#> Athlete 05 Treatment 56.88816  57.73773  0.84957368
#> Athlete 06 Control  52.31237  51.82011 -0.49226674
#> Athlete 07 Treatment 54.93702  55.28749  0.35047044
#> Athlete 08 Control  51.29489  51.15000 -0.14488443
#> Athlete 09 Treatment 63.26964  60.38212 -2.88752039
#> Athlete 10 Control  50.28937  49.38628 -0.90309058
#> Athlete 11 Treatment 59.39720  60.04667  0.64947349
#> Athlete 12 Control  50.02810  50.06914  0.04103265
#> Athlete 13 Treatment 55.63766  54.34070 -1.29696009
#> Athlete 14 Control  46.82518  48.34585  1.52066587
#> Athlete 15 Treatment 57.89749  56.09115 -1.80634311
#> Athlete 16 Control  50.56725  50.45161 -0.11564089
#> Athlete 17 Treatment 55.12290  59.00610  3.88319663
#> Athlete 18 Control  48.30309  49.24440  0.94130869
#> Athlete 19 Treatment 56.22363  56.47776  0.25412115
#> Athlete 20 Control  51.21115  48.84901 -2.36213638
#>      magnitude counterfactual      pITE pITE_magnitude
#> Equivalent      46.93258 -8.004564           Lower
#> Equivalent      56.49986  6.344157           Higher
#> Equivalent      49.34023 -9.714660          Lower
#> Equivalent      54.38014  2.705324          Equivalent
#> Equivalent      47.88067 -9.857066          Lower
#> Equivalent      57.21781  5.397698           Higher
#> Equivalent      51.37291 -3.914577          Equivalent
#> Equivalent      58.01547  6.865468           Higher

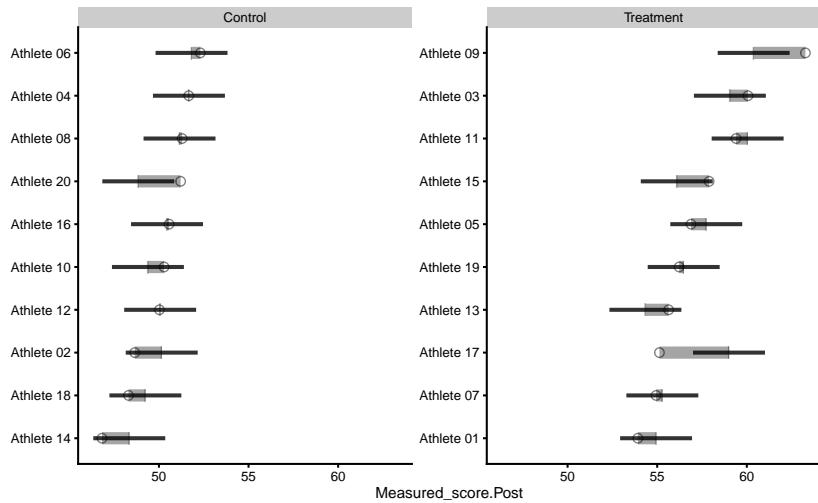
```

```

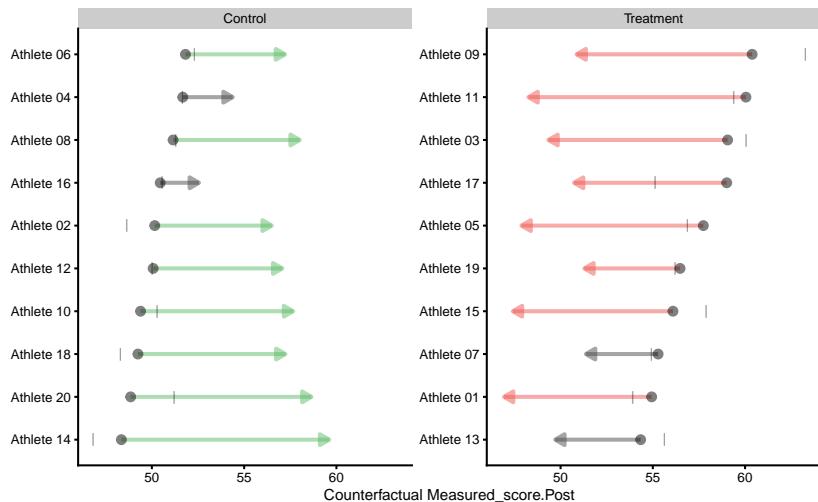
#>   Equivalent      50.86403 -9.518092      Lower
#>   Equivalent      57.65772  8.271448      Higher
#>   Equivalent      48.26002 -11.786649     Lower
#>   Equivalent      57.06806  6.998928      Higher
#>   Equivalent      49.70900 -4.631703    Equivalent
#>   Equivalent      59.61112 11.265266      Higher
#>   Equivalent      47.40976 -8.681390     Lower
#>   Equivalent      52.55310  2.101491    Equivalent
#>   Equivalent      50.72701 -8.279084     Lower
#>   Equivalent      57.23888  7.994484      Higher
#>   Equivalent      51.28511 -5.192643     Lower
#>   Equivalent      58.65050  9.801492      Higher
#>
#> Summary of residuals per RCT group:
#>
#>   group       mean        SD
#>   Control 7.105211e-16 1.161243
#>   Treatment 4.263218e-15 1.878159
#>
#> Summary of counterfactual effects of RCT group:
#>
#>   group      pATE      pVTE
#>   Treatment -7.958043 2.570647
#>   Control   6.774576 2.859555
#>   pooled    7.366309 2.715167
#>
#> Treatment effect summary
#>
#> Average Treatment effect: 7.366309
#> Variable Treatment effect: 2.715167
#> Random Treatment effect: 1.476142

```

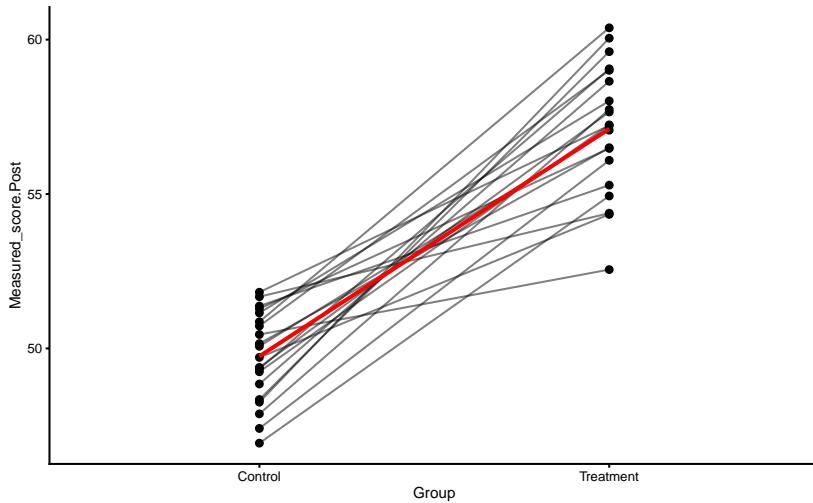
```
plot(interaction_model, "prediction")
```



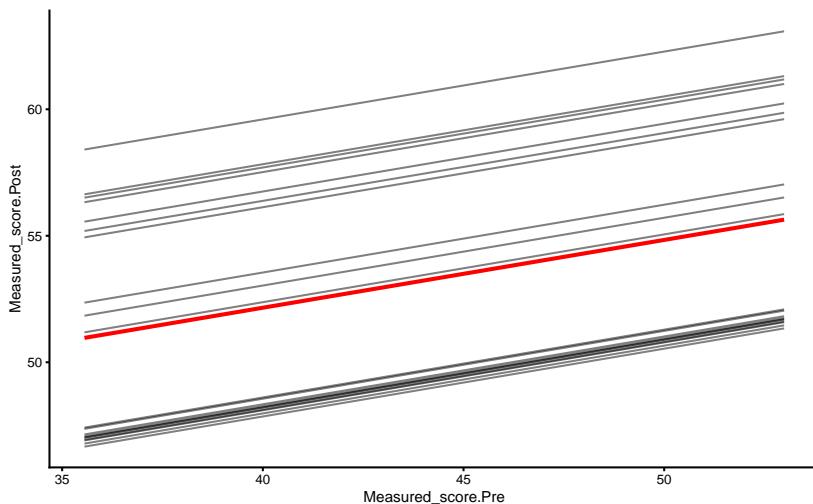
```
plot(interaction_model, "counterfactual")
```



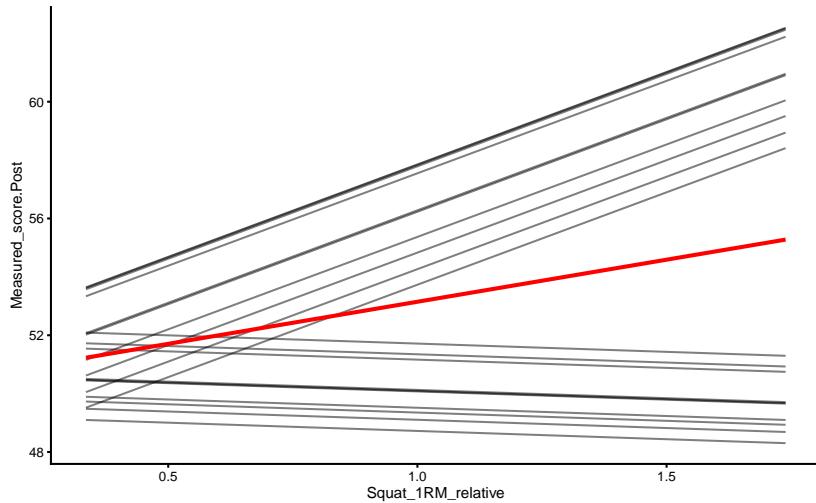
```
plot(interaction_model, "pdp+ice")
```



```
plot(interaction_model, "pdp+ice", predictor = "Measured_score.Pre")
```



```
plot(interaction_model, "pdp+ice", predictor = "Squat_1RM_relative")
```



For the sake of comparison between the models, let's pull out estimated average, random, and variable treatment effects. Average treatment effect (pATE) is estimated using the `mean` of the pooled (absolute) counterfactual effects (see the previous model print summaries). Variable treatment effect (pVTE) is estimated using the `SD` of the pooled (absolute) counterfactual effects. Random treatment effect (RTE) is estimated using the group residuals as explained thorough this chapter.

```
model_estimates <- rbind(data.frame(model = "base", pATE = base_model$extra$average_treatment_effect,
                                     pVTE = base_model$extra$variable_treatment_effect, RTE = base_model$extra$random_treatment_effect),
                           data.frame(model = "pre-test", pATE = pre_test_model$extra$average_treatment_effect,
                                      pVTE = pre_test_model$extra$variable_treatment_effect,
                                      RTE = pre_test_model$extra$random_treatment_effect),
                           data.frame(model = "covariate", pATE = covariate_model$extra$average_treatment_effect,
                                      pVTE = covariate_model$extra$variable_treatment_effect,
                                      RTE = covariate_model$extra$random_treatment_effect),
                           data.frame(model = "interaction", pATE = interaction_model$extra$average_treatment_effect,
                                      pVTE = interaction_model$extra$variable_treatment_effect,
                                      RTE = interaction_model$extra$random_treatment_effect))

model_estimates
#>       model      pATE        pVTE        RTE
#> 1     base 7.221484 0.000000e+00 2.298433
#> 2   pre-test 7.599370 3.992905e-15 2.521028
#> 3 covariate 7.421514 4.890290e-15 1.599750
#> 4 interaction 7.366309 2.715167e+00 1.476142
```

Using counterfactual prediction, which is estimated by changing the group (and also group interaction with the squat 1RM), allows us to estimate average

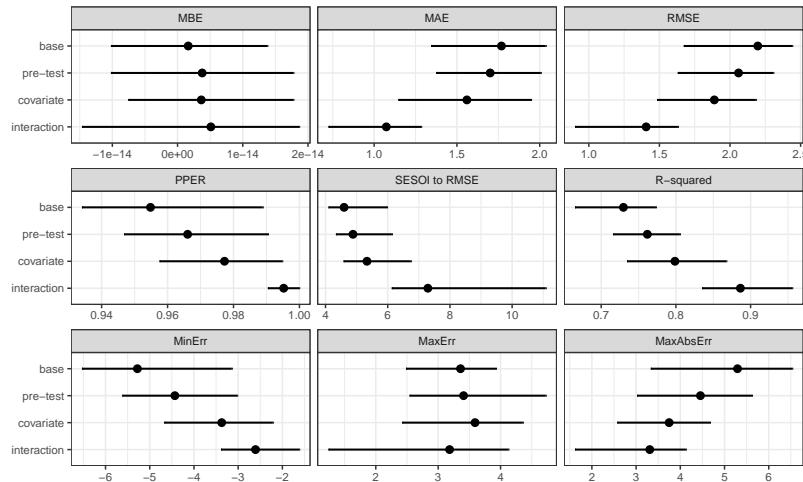
treatment effect (i.e. systematic or expected treatment effect) from the model with interactions.

Let's plot the model predictive performance to compare the models. Let's check the training performance (using training CV folds, where error bars represent min and max across training folds):

```
model_performace <- rbind(data.frame(model = "base", base_model$cross_validation$performance$summary,
                                       data.frame(model = "pre-test", pre_test_model$cross_validation$performance$summary$training),
                                       data.frame(model = "covariate", covariate_model$cross_validation$performance$summary$training),
                                       data.frame(model = "interaction", interaction_model$cross_validation$performance$summary$training))

model_performace$model <- factor(model_performace$model,
                                   levels = rev(c("base", "pre-test", "covariate", "interaction")))

ggplot(model_performace, aes(y = model, x = mean)) + theme_bw(8) +
  geom_errorbarh(aes(xmax = min, xmin = max), color = "black",
                 height = 0) + geom_point() + xlab("") + ylab("") +
  facet_wrap(~metric, scales = "free_x")
```

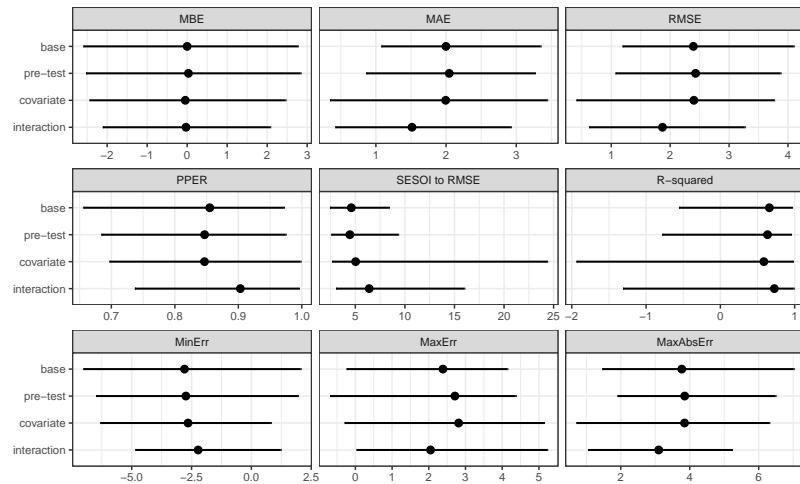


As can be seen from the figure, interaction model has the best training folds predictive performance. What about performance on the testing CV folds?

```
model_performace <- rbind(data.frame(model = "base", base_model$cross_validation$performance$summary,
                                       data.frame(model = "pre-test", pre_test_model$cross_validation$performance$summary$testing),
                                       data.frame(model = "covariate", covariate_model$cross_validation$performance$summary$testing),
                                       data.frame(model = "interaction", interaction_model$cross_validation$performance$summary$testing))
```

```
model_performace$model <- factor(model_performace$model,
  levels = rev(c("base", "pre-test", "covariate", "interaction")))

ggplot(model_performace, aes(y = model, x = mean)) + theme_bw(8) +
  geom_errorbarh(aes(xmax = min, xmin = max), color = "black",
  height = 0) + geom_point() + xlab("") + ylab("") +
  facet_wrap(~metric, scales = "free_x")
```



Interaction model is better, but not drastically better. Mean testing PPER is pretty good, over 0.9 indicating good practical predictive performance of this model.

Although the simple differences method or linear regression model helped us to *explain* overall treatment effects, predictive approach can tell us if we are able to predict individual responses, but also help in interpreting model counterfactually, particularly when there is interaction terms in the model.

We have used linear regression as our predictive model, but we could use any other model for that matter. Having RCT design allows us to interpret certain effect causally. Discussing these topics further is beyond the scope of this book (and the author) and the reader is directed to the references provided.

Chapter 17

Appendix A: `dorem` package

The goal of `dorem` (Jovanovic and Hemingway 2020) is to provide easy-to-use dose-response models utilized in sport science. This package is currently in active development phases.

17.1 `dorem` Installation

You can install the development version from GitHub¹ with:

```
# install.packages("devtools")
devtools::install_github("mladenjovanovic/dorem")

require(dorem)
```

17.2 `dorem` Example

To provide very simplistic example of `dorem`, I will use example data provided in supplementary material² of Clarke & Skiba, 2013 paper³ (Clarke and Skiba 2013), freely available on the publisher website. Data set contains cycling training load (i.e. dose) measured using the BikeScore metric (in AU) over 165 days, with occasional training response measured using 5-min Power Test (in Watts). *Banister* model (explained in aforementioned paper) is applied to understand relationship between *training dose* (i.e., BikeScore metric) and *training response* (i.e., 5-min Power Test):

¹<https://github.com/mladenjovanovic/dorem>

²<https://journals.physiology.org/doi/full/10.1152/advan.00078.2011m>

³<https://journals.physiology.org/doi/full/10.1152/advan.00078.2011?>

```

require(dorem)
require(tidyverse)
require(cowplot)

data("bike_score")

banister_model <- dorem(Test_5min_Power ~ BikeScore, bike_score,
  method = "banister")

# Print results
banister_model
#> Dose-Response Model using banister method
#> Training data consists of 1 predictor and 165 observations
#> Coefficients are estimated using L-BFGS-B method with 1000 max iterations and -Inf
#>
#> The following start and bound values were used:
#>
#>          start lower upper
#> intercept      271     0   302
#> BikeScore.PTE_gain    1     0 10000
#> BikeScore.PTE_tau     21     0   300
#> BikeScore.NTE_gain    3     0 10000
#> BikeScore.NTE_tau     7     0   300
#>
#> Estimated model coefficients are the following:
#>
#>          intercept BikeScore.PTE_gain
#>        266.0204779      0.3020749
#> BikeScore.PTE_tau BikeScore.NTE_gain
#>        33.6433797      0.3514750
#> BikeScore.NTE_tau
#>        24.6056388
#>
#> Objective function equal to: 2.1939
#>
#> Cross-Validation of the model was not performed. Shuffling of the predictors was no
#>
#> Overall model performance using selected estimators is the following:
#>
#>          training
#> N         9.00000000
#> meanDiff  0.01374013
#> SDdiff    1.57096328
#> RMSE      1.48118212
#> MAE       1.16731419

```

```

#> minErr      -2.71352646
#> maxErr      2.22332820
#> MAPE        0.40904959
#> R_squared   0.97975692

# get coeffs
coef(banister_model)
#>           intercept BikeScore.PTE_gain
#>       266.0204779      0.3020749
#> BikeScore.PTE_tau BikeScore.NTE_gain
#>       33.6433797      0.3514750
#> BikeScore.NTE_tau
#>       24.6056388

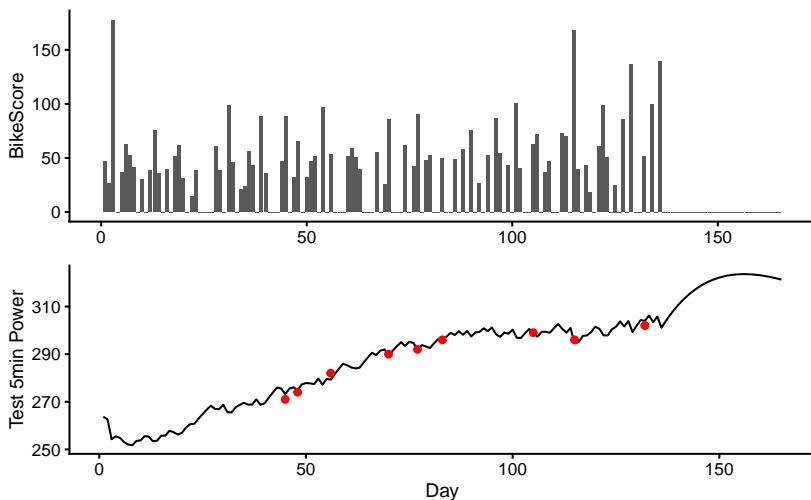
# Get model predictions
bike_score$pred <- predict(banister_model, bike_score)$pred

# Plot
dose <- ggplot(bike_score, aes(x = Day, y = BikeScore)) +
  theme_cowplot(10) + geom_bar(stat = "identity") + xlab(NULL)

response <- ggplot(bike_score, aes(x = Day, y = pred)) +
  theme_cowplot(10) + geom_line() + geom_point(aes(y = Test_5min_Power),
  color = "red") + ylab("Test 5min Power")

cowplot:::plot_grid(dose, response, ncol = 1)

```



`dorem` also allows more control and setup using the `control` parameter. In

the next example, cross-validation of 3 repeats and 5 folds will be performed, with additional feature of *shuffling* the predictors and evaluating how the model predicts on random predictors (i.e., dose):

```
banister_model <- dorem(
  Test_5min_Power ~ BikeScore,
  bike_score,
  method = "banister",

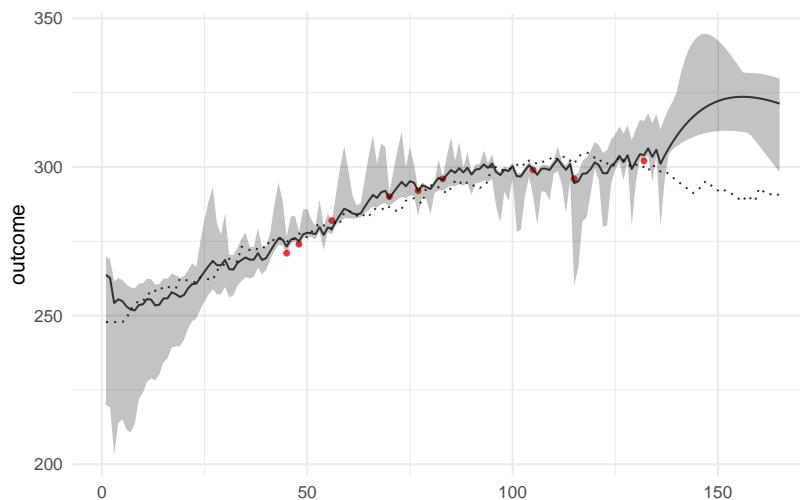
  # control setup
  control = dorem_control(
    shuffle = TRUE,
    optim_method = "L-BFGS-B",
    optim_maxit = 1000,
    cv_folds = 3,
    cv_repeats = 5
  )
)

banister_model
#> Dose-Response Model using banister method
#> Training data consists of 1 predictor and 165 observations
#> Coefficients are estimated using L-BFGS-B method with 1000 max iterations and -Inf
#>
#> The following start and bound values were used:
#>
#>           start lower upper
#> intercept      271     0   302
#> BikeScore.PTE_gain    1     0 10000
#> BikeScore.PTE_tau     21     0   300
#> BikeScore.NTE_gain     3     0 10000
#> BikeScore.NTE_tau     7     0   300
#>
#> Estimated model coefficients are the following:
#>
#>           intercept BikeScore.PTE_gain
#>           266.0204779      0.3020749
#> BikeScore.PTE_tau BikeScore.NTE_gain
#>           33.6433797      0.3514750
#> BikeScore.NTE_tau
#>           24.6056388
#>
#> Objective function equal to: 2.1939
#>
#> Cross-Validation of the model was performed using 5 repeats of 3 folds. Shuffling o
```

```
#> Overall model performance using selected estimators is the following:
#>
#>           training          CV    shuffle
#> N      9.00000000 45.0000000 9.0000000
#> meanDiff 0.01374013 -1.4468364 -0.0149063
#> SDdiff 1.57096328 6.5352444 3.8231697
#> RMSE   1.48118212 6.6222094 3.6045497
#> MAE    1.16731419 3.8249026 3.5302780
#> minErr -2.71352646 -35.6767605 -3.9240093
#> maxErr 2.22332820 13.4489856 4.9477488
#> MAPE   0.40904959 1.3119052 1.2250645
#> R_squared 0.97975692 0.6838326 0.8800912
```

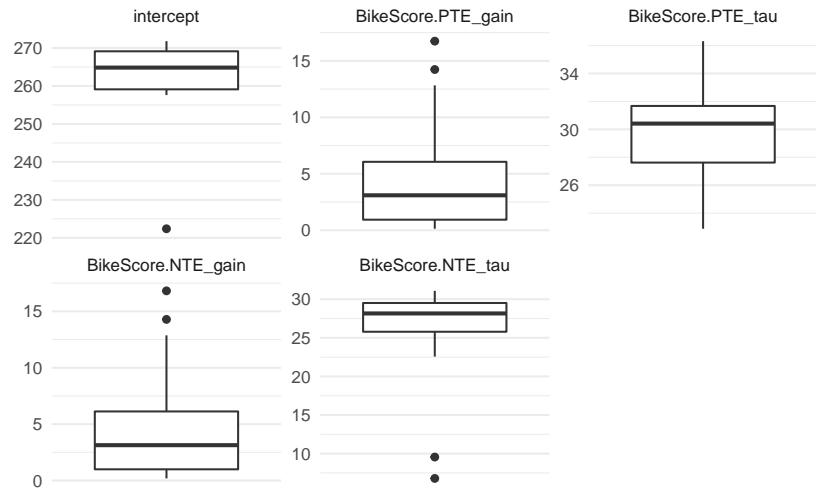
To plot model predictions, including the CV as gray area and shuffle as dotted line, use:

```
plot(banister_model, type = "pred") + theme_minimal()
```



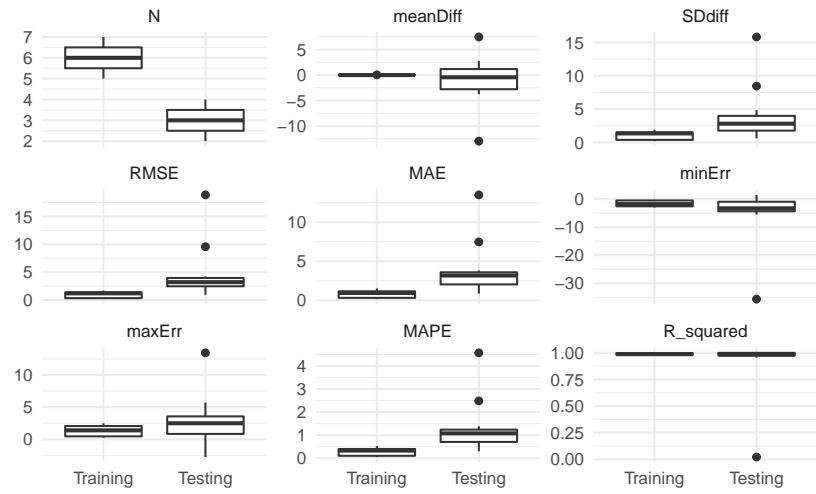
To plot model coefficients across CV folds:

```
plot(banister_model, type = "coef") + theme_minimal()
```



To plot model performance across CV folds (i.e., training and testing folds):

```
plot(banister_model, type = "perf") + theme_minimal()
```



Chapter 18

Appendix B: `shorts` package

Package `short` (Jovanovic 2020) creates short sprint (<6sec) profiles using the split times or the radar gun data. Mono-exponential equation is used to estimate maximal sprinting speed (MSS), relative acceleration (TAU), and other parameters. These parameters can be used to predict kinematic and kinetics variables and to compare individuals.

18.1 `shorts` Installation

```
# Install from CRAN
install.packages("shorts")

# Or the development version from GitHub
# install.packages("devtools")
devtools::install_github("mladenjovanovic/shorts")
```

18.2 `short` Examples

`shorts` comes with two sample data sets: `split_times` and `radar_gun_data` with N=5 athletes. Let's load them both:

```
require(shorts)
require(tidyverse)
```

```
data("split_times", "radar_gun_data")
```

18.2.1 Profiling using split times

To model sprint performance using split times, distance will be used as predictor and time as target. Since `split_times` contains data for multiple athletes, let's extract only one athlete and model it using `shorts::model_using_splits` function.

```
kimberley_data <- filter(split_times, athlete == "Kimberley")

kimberley_data
#> # A tibble: 6 x 4
#>   athlete   bodyweight distance     time
#>   <chr>        <dbl>    <dbl> <I<dbl>>
#> 1 Kimberley      55       5    1.16
#> 2 Kimberley      55      10    1.89
#> 3 Kimberley      55      15    2.54
#> 4 Kimberley      55      20    3.15
#> 5 Kimberley      55      30    4.31
#> 6 Kimberley      55      40    5.44
```

`shorts::model_using_splits` returns an object with `parameters`, `model_fit`, `model` returned from `stats::nls` function and `data` used to estimate parameters. Parameters estimated using mono-exponential equation are *maximal sprinting speed* (MSS), and *relative acceleration* (TAU). Additional parameters computed from MSS and TAU are *maximal acceleration* (MAC) and *maximal relative power* (PMAX).

```
kimberley_profile <- shorts::model_using_splits(distance = kimberley_data$distance,
                                                time = kimberley_data$time)

kimberley_profile
#> Estimated model parameters
#> -----
#>          MSS             TAU
#>          8.5911421        0.8113282
#>          MAC             PMAX
#>          10.5889855       22.7428698
#>          time_correction distance_correction
#>          0.0000000        0.0000000
#>
#> Model fit estimators
```

```

#> -----
#>      RSE   R_squared      minErr      maxErr
#> 0.03403413 0.99965531 -0.02699169 0.05293444
#> maxAbsErr      RMSE       MAE       MAPE
#> 0.05293444 0.02778875 0.02333342 1.19263116

summary(kimberley_profile)
#>
#> Formula: corrected_time ~ TAU * I(LambertW::W(-exp(1)^(-distance/(MSS *
#>           TAU) - 1))) + distance/MSS + TAU
#>
#> Parameters:
#>   Estimate Std. Error t value Pr(>|t|)
#> MSS 8.59114     0.12251    70.13 2.48e-07 ***
#> TAU 0.81133     0.04581    17.71 5.97e-05 ***
#> ---
#> Signif. codes:
#> 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 0.03403 on 4 degrees of freedom
#>
#> Number of iterations to convergence: 4
#> Achieved convergence tolerance: 4.058e-06

coef(kimberley_profile)
#>          MSS             TAU
#> 8.5911421     0.8113282
#>          MAC             PMAX
#> 10.5889855    22.7428698
#> time_correction distance_correction
#> 0.0000000     0.0000000

```

To return the predicted outcome (in this case time variable), use `predict` function:

```

predict(kimberley_profile)
#> [1] 1.210934 1.897021 2.521028 3.122008 4.299243
#> [6] 5.466325

```

If you are interested in calculating average split velocity, use `shorts::format_splits`

```

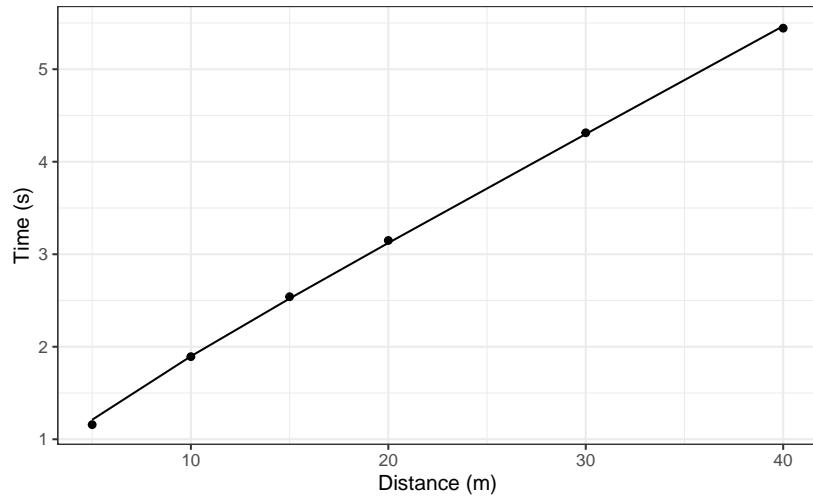
shorts::format_splits(distance = kimberley_data$distance,
                      time = kimberley_data$time)
#>   split split_distance_start split_distance_stop
#> 1      1                  0                   5

```

```
#> 2      2          5          10
#> 3      3          10         15
#> 4      4          15         20
#> 5      5          20         30
#> 6      6          30         40
#>   split_distance split_time_start split_time_stop
#> 1      5           0        1.158
#> 2      5          1.158      1.893
#> 3      5          1.893      2.541
#> 4      5          2.541      3.149
#> 5      10         3.149      4.313
#> 6      10         4.313      5.444
#>   split_time split_mean_velocity
#> 1      1.158      4.317789....
#> 2      0.735      6.802721....
#> 3      0.648      7.716049....
#> 4      0.608      8.223684....
#> 5      1.164      8.591065....
#> 6      1.131      8.841732....
```

Let's plot observed vs fitted split times. For this we can use `data` returned from `shorts::model_using_splits` since it contains `pred_time` column.

```
ggplot(kimberley_profile$data, aes(x = distance)) + theme_bw() +
  geom_point(aes(y = time)) + geom_line(aes(y = pred_time)) +
  xlab("Distance (m)") + ylab("Time (s)")
```



To plot predicted velocity, acceleration, and relative power over distance, use `shorts::predict_`

```
kimberley_pred <- tibble(
  distance = seq(0, 40, length.out = 1000),

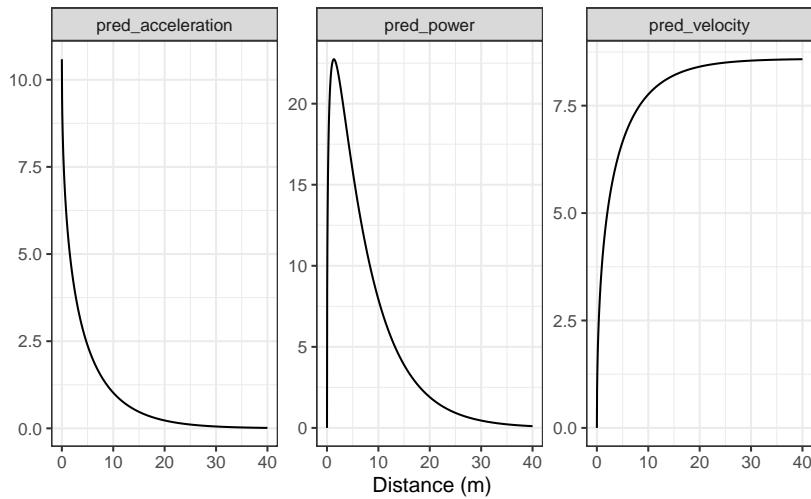
  # Velocity
  pred_velocity = shorts::predict_velocity_at_distance(
    distance,
    kimberley_profile$parameters$MSS,
    kimberley_profile$parameters$TAU),

  # Acceleration
  pred_acceleration = shorts::predict_acceleration_at_distance(
    distance,
    kimberley_profile$parameters$MSS,
    kimberley_profile$parameters$TAU),

  # Power
  pred_power = shorts::predict_relative_power_at_distance(
    distance,
    kimberley_profile$parameters$MSS,
    kimberley_profile$parameters$TAU),
)

# Convert to long
kimberley_pred <- gather(kimberley_pred, "metric", "value", -distance)

ggplot(kimberley_pred, aes(x = distance, y = value)) +
  theme_bw() +
  geom_line() +
  facet_wrap(~metric, scales = "free_y") +
  xlab("Distance (m)") +
  ylab(NULL)
```



To do prediction simpler, use `shorts::predict_kinematics` function. This will provide kinematics for 0-6s sprint using 100Hz.

```
predicted_kinematics <- predict_kinematics(kimberley_profile)
head(predicted_kinematics)
#>   time      distance    velocity acceleration      power
#> 1 0.00 0.0000000000 0.0000000 10.588986 0.000000
#> 2 0.01 0.0005272807 0.1052400 10.459272 1.100733
#> 3 0.02 0.0021005019 0.2091907 10.331148 2.161181
#> 4 0.03 0.0047068510 0.3118682 10.204593 3.182488
#> 5 0.04 0.0083336724 0.4132878 10.079589 4.165771
#> 6 0.05 0.0129684654 0.5134650 9.956116 5.112117
```

To get model residuals, use `residuals` function:

```
residuals(kimberley_profile)
#> [1] 0.052934436 0.004021074 -0.019971823 -0.026991691
#> [5] -0.013756850 0.022324628
```

Package `shorts` comes with `find_` family of functions that allow finding peak power and it's location, as well as *critical distance* over which velocity, acceleration, or power drops below certain threshold:

```
# Peak power and location
shorts::find_max_power_distance(kimberley_profile$parameters$MSS,
                                 kimberley_profile$parameters$TAU)
#> $max_power
```

```

#> [1] 22.74287
#>
#> $distance
#> [1] 1.346271

# Distance over which power is over 50%
shorts::find_power_critical_distance(MSS = kimberley_profile$parameters$MESS,
                                       TAU = kimberley_profile$parameters$TAU, percent = 0.5)
#> $lower
#> [1] 0.08295615
#>
#> $upper
#> [1] 7.441024

# Distance over which acceleration is under 50%
shorts::find_acceleration_critical_distance(MSS = kimberley_profile$parameters$MESS,
                                               TAU = kimberley_profile$parameters$TAU, percent = 0.5)
#> [1] 1.346279

# Distance over which velocity is over 95%
shorts::find_velocity_critical_distance(MSS = kimberley_profile$parameters$MESS,
                                         TAU = kimberley_profile$parameters$TAU, percent = 0.95)
#> [1] 14.25922

```

18.2.1.1 Mixed-effect models

Each individual can be modeled separately, or we can perform *non-linear mixed model* using `nlme` function from `nlme` package (Pinheiro *et al.*, 2019). This is done using `shorts::mixed_model_using_splits`:

```

mixed_model <- shorts::mixed_model_using_splits(data = split_times,
                                                distance = "distance", time = "time", athlete = "athlete")

mixed_model
#> Estimated fixed model parameters
#> -----
#>          MSS             TAU
#>     8.0649112      0.6551988
#>          MAC             PMAX
#>    12.3091052      24.8179600
#> time_correction distance_correction
#>      0.0000000      0.0000000
#>
#> Estimated frandom model parameters

```

```

#> -----
#>     athlete      MSS       TAU       MAC      PMAX
#> 1   James 9.691736 0.8469741 11.44278 27.72510
#> 2   Jim 7.833622 0.5048535 15.51663 30.38785
#> 3   John 7.780395 0.7274302 10.69573 20.80424
#> 4 Kimberley 8.569518 0.8022235 10.68221 22.88535
#> 5 Samantha 6.449284 0.3945129 16.34746 26.35735
#>   time_correction distance_correction
#> 1           0           0
#> 2           0           0
#> 3           0           0
#> 4           0           0
#> 5           0           0
#>
#> Model fit estimators
#> -----
#>     RSE    R_squared      minErr      maxErr
#> 0.02600213 0.99982036 -0.02934519 0.04964582
#> maxAbsErr      RMSE      MAE      MAPE
#> 0.04964582 0.02139178 0.01722581 0.90185579

summary(mixed_model)
#> Nonlinear mixed-effects model fit by maximum likelihood
#> Model: corrected_time ~ TAU * I(LambertW::W(-exp(1)^(-distance/(MSS *
#> Data: train
#>          AIC      BIC logLik
#> -75.06719 -66.66001 43.5336
#>
#> Random effects:
#> Formula: list(MSS ~ 1, TAU ~ 1)
#> Level: athlete
#> Structure: General positive-definite, Log-Cholesky parametrization
#>          StdDev     Corr
#> MSS      1.06581655 MSS
#> TAU      0.17821114 0.877
#> Residual 0.02600213
#>
#> Fixed effects: MSS + TAU ~ 1
#>          Value Std.Error DF t-value p-value
#> MSS 8.064911 0.4949104 24 16.295699      0
#> TAU 0.655199 0.0837593 24  7.822404      0
#> Correlation:
#>          MSS
#> TAU 0.874
#>

```

```
#> Standardized Within-Group Residuals:
#>      Min       Q1      Med       Q3      Max
#> -1.9092981 -0.6050683  0.1536529  0.5226467  1.1285687
#>
#> Number of Observations: 30
#> Number of Groups: 5

coef(mixed_model)
#> $fixed
#>           MSS          TAU
#> 8.0649112 0.6551988
#>          MAC          PMAX
#> 12.3091052 24.8179600
#> time_correction distance_correction
#> 0.0000000 0.0000000
#>
#> $random
#>   athlete    MSS      TAU      MAC      PMAX
#> 1 James 9.691736 0.8469741 11.44278 27.72510
#> 2 Jim 7.833622 0.5048535 15.51663 30.38785
#> 3 John 7.780395 0.7274302 10.69573 20.80424
#> 4 Kimberley 8.569518 0.8022235 10.68221 22.88535
#> 5 Samantha 6.449284 0.3945129 16.34746 26.35735
#> time_correction distance_correction
#> 1 0 0
#> 2 0 0
#> 3 0 0
#> 4 0 0
#> 5 0 0
```

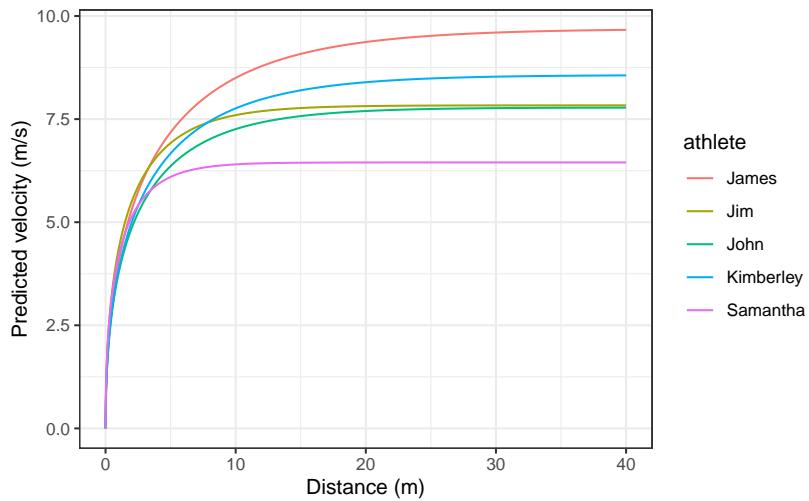
`shorts::mixed_model_using_splits` return the similar object, but parameters contain two elements: `fixed` and `random`.

Let's plot predicted velocity over distance for athletes in the `split_times` data set:

```
velocity_over_distance <- merge(mixed_model$parameters$random,
  data.frame(distance = seq(0, 40, length.out = 1000)))

velocity_over_distance$pred_velocity <- with(velocity_over_distance,
  shorts::predict_velocity_at_distance(distance = distance,
  MSS = MSS, TAU = TAU))

ggplot(velocity_over_distance, aes(x = distance, y = pred_velocity,
  color = athlete)) + theme_bw() + geom_line() + xlab("Distance (m)") +
  ylab("Predicted velocity (m/s)")
```



To modify random effects, which are by default MSS and TAU ($MSS + TAU \sim 1$), use the `random` parameter. For example, we can assume same TAU for all athletes and only use MSS as random effect:

```
mixed_model <- shorts::mixed_model_using_splits(data = split_times,
  distance = "distance", time = "time", athlete = "athlete",
  random = MSS ~ 1)

mixed_model
#> Estimated fixed model parameters
#> -----
#>          MSS           TAU
#>    7.9366665      0.6277251
#>          MAC           PMAX
#>   12.6435385     25.0868872
#> time_correction distance_correction
#>    0.0000000      0.0000000
#>
#> Estimated frandom model parameters
#> -----
#>   athlete   MSS      TAU      MAC      PMAX
#> 1   James 9.021822 0.6277251 14.37225 32.41597
#> 2   Jim 8.111530 0.6277251 12.92211 26.20451
#> 3   John 7.576142 0.6277251 12.06920 22.85950
#> 4 Kimberley 8.144414 0.6277251 12.97449 26.41741
#> 5 Samantha 6.829424 0.6277251 10.87964 18.57542
#> time_correction distance_correction
#> 1             0            0
```

```
#> 2          0          0
#> 3          0          0
#> 4          0          0
#> 5          0          0
#>
#> #> Model fit estimators
#> -----
#>      RSE   R_squared   minErr   maxErr
#> 0.07635631 0.99801628 -0.10228573 0.15987989
#> maxAbsErr      RMSE      MAE      MAPE
#> 0.15987989 0.06979851 0.05845303 2.68555639
```

18.2.2 Profiling using radar gun data

The radar gun data is modeled using measured velocity as target variable and time as predictor. Individual analysis is performed using `shorts::model_using_radar` function. Let's do analysis for Jim:

```
jim_data <- filter(radar_gun_data, athlete == "Jim")

jim_profile <- shorts::model_using_radar(time = jim_data$time,
                                           velocity = jim_data$velocity)

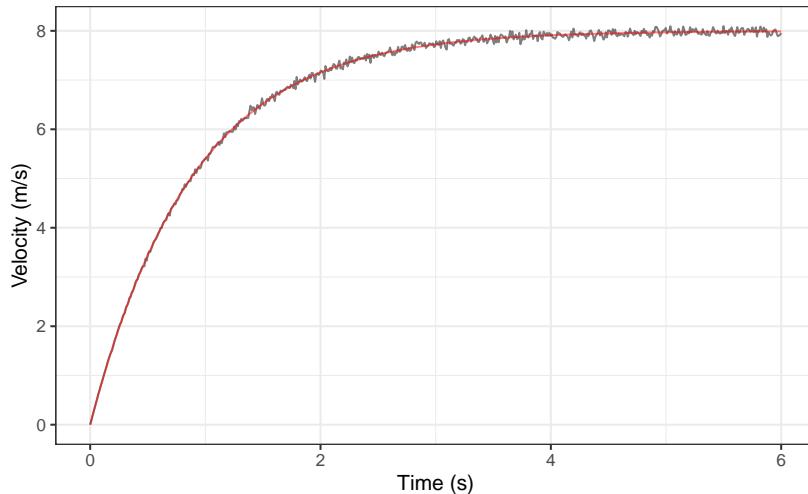
jim_profile
#> #> Estimated model parameters
#> -----
#>      MSS           TAU
#> 7.9979331 0.8886595
#> MAC           PMAX
#> 8.9999977 17.9953449
#> time_correction distance_correction
#> 0.0000000 0.0000000
#>
#> #> Model fit estimators
#> -----
#>      RSE   R_squared   minErr   maxErr
#> 0.05058726 0.99924408 -0.15099212 0.16415830
#> maxAbsErr      RMSE      MAE      MAPE
#> 0.16415830 0.05050288 0.03927901      NaN

summary(jim_profile)
#>
#> Formula: velocity ~ MSS * (1 - exp(1)^(-(corrected_time)/TAU))
#>
```

```
#> Parameters:
#>   Estimate Std. Error t value Pr(>|t|)
#> MSS 7.997933  0.003069 2606.3 <2e-16 ***
#> TAU 0.888659  0.001564   568.2 <2e-16 ***
#> ---
#> Signif. codes:
#> 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 0.05059 on 598 degrees of freedom
#>
#> Number of iterations to convergence: 3
#> Achieved convergence tolerance: 9.313e-07
```

The object returned from `shorts::model_using_radar` is same as object returned from `shorts::model_using_splits`. Let's plot Jim's measured velocity and predicted velocity:

```
ggplot(jim_profile$data, aes(x = time)) + theme_bw() + geom_line(aes(y = velocity),
alpha = 0.5) + geom_line(aes(y = pred_velocity), color = "red",
alpha = 0.5) + xlab("Time (s)") + ylab("Velocity (m/s)")
```



Radar gun data can be modeled individually or using *non-linear mixed model* implemented in `shorts::mixed_model_using_radar`:

```
mixed_model <- shorts::mixed_model_using_radar(data = radar_gun_data,
time = "time", velocity = "velocity", athlete = "athlete")
```

```

mixed_model
#> Estimated fixed model parameters
#> -----
#>          MSS           TAU
#> 8.301178      1.007782
#>          MAC           PMAX
#> 8.237080      17.094367
#> time_correction distance_correction
#> 0.000000      0.000000
#>
#> Estimated frandom model parameters
#> -----
#>   athlete    MSS     TAU     MAC     PMAX
#> 1 James 9.998556 1.1108457 9.000851 22.49888
#> 2 Jim 7.997945 0.8886712 8.999892 17.99516
#> 3 John 8.000051 1.0690357 7.483427 14.96695
#> 4 Kimberley 9.005500 1.2855706 7.005061 15.77102
#> 5 Samantha 6.503839 0.6847851 9.497635 15.44277
#> time_correction distance_correction
#> 1          0          0
#> 2          0          0
#> 3          0          0
#> 4          0          0
#> 5          0          0
#>
#> Model fit estimators
#> -----
#>       RSE   R_squared      minErr      maxErr
#> 0.05164818 0.99942171 -0.21912952 0.19832897
#> maxAbsErr      RMSE      MAE      MAPE
#> 0.21912952 0.05156203 0.03949473      NaN

summary(mixed_model)
#> Nonlinear mixed-effects model fit by maximum likelihood
#> Model: velocity ~ MSS * (1 - exp(1)^(-(corrected_time)/TAU))
#> Data: train
#>       AIC      BIC logLik
#> -9150.177 -9114.139 4581.089
#>
#> Random effects:
#> Formula: list(MSS ~ 1, TAU ~ 1)
#> Level: athlete
#> Structure: General positive-definite, Log-Cholesky parametrization
#>          StdDev   Corr
#> MSS      1.16535852 MSS

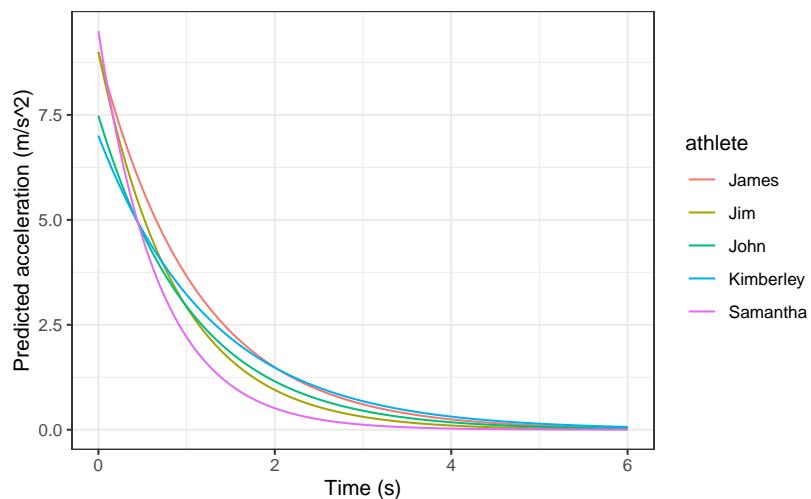
```

```
#> TAU      0.20497379 0.811
#> Residual 0.05164818
#>
#> Fixed effects: MSS + TAU ~ 1
#>           Value Std.Error DF t-value p-value
#> MSS 8.301178 0.5213403 2994 15.92276      0
#> TAU 1.0077782 0.0917011 2994 10.98986      0
#> Correlation:
#>   MSS
#>   TAU 0.811
#>
#> Standardized Within-Group Residuals:
#>   Min            Q1            Med            Q3
#> -3.8399995748 -0.5932967650 -0.0002562518  0.6111624765
#>   Max
#>  4.2427350284
#>
#> Number of Observations: 3000
#> Number of Groups: 5
```

Let's plot predicted acceleration over time (0-6sec) for athletes in the `radar_gun_data` data set:

```
model_predictions <- shorts::predict_kinematics(mixed_model)

ggplot(model_predictions, aes(x = time, y = acceleration,
  color = athlete)) + theme_bw() + geom_line() + xlab("Time (s)") +
  ylab("Predicted acceleration (m/s^2)")
```



18.2.3 Using corrections

You have probably noticed that estimated MSS and TAU were a bit too high for splits data. Biased estimates are due to differences in starting positions and *timing triggering methods* for certain measurement approaches (e.g. starting behind first timing gate, or allowing for body rocking). This topic is further explained in `sprint-corrections` vignette¹ that can be accessed by typing:

```
vignette("sprint-corrections")
```

Here I will provide quick summary. Often, this bias in estimates is dealt with by using heuristic rule of thumb of adding `time_correction` to split times (e.g. from 0.3–0.5sec; see more in Haugen *et al.*, 2012). This functionality is available in all covered `shorts` functions:

```
mixed_model_corrected <- shorts::mixed_model_using_splits(data = split_times,
  distance = "distance", time = "time", athlete = "athlete",
  time_correction = 0.3)

mixed_model_corrected
#> Estimated fixed model parameters
#> -----
#>          MSS           TAU
#> 8.474621 1.154940
#>          MAC           PMAX
#> 7.337715 15.546088
#> time_correction distance_correction
#> 0.300000 0.000000
#>
#> Estimated frandom model parameters
#> -----
#>   athlete    MSS      TAU      MAC      PMAX
#> 1 James 10.549314 1.4953619 7.054689 18.60553
#> 2 Jim 8.048378 0.9216038 8.733012 17.57165
#> 3 John 8.130968 1.2295728 6.612839 13.44220
#> 4 Kimberley 9.114979 1.3721302 6.642940 15.13756
#> 5 Samantha 6.529465 0.7560310 8.636504 14.09794
#> time_correction distance_correction
#> 1 0.3 0
#> 2 0.3 0
#> 3 0.3 0
#> 4 0.3 0
#> 5 0.3 0
#>
```

¹<https://mladenjovanovic.github.io/shorts/articles/sprint-corrections.html>

```

#> Model fit estimators
#> -----
#>      RSE    R_squared      minErr      maxErr
#> 0.015195052 0.999941466 -0.041155421 0.020298042
#> maxAbsErr      RMSE       MAE       MAPE
#> 0.041155421 0.012443740 0.009087699 0.496822694

summary(mixed_model_corrected)
#> Nonlinear mixed-effects model fit by maximum likelihood
#> Model: corrected_time ~ TAU * I(LambertW::W(-exp(1)^(-distance/(MSS *
#> Data: train
#>      AIC      BIC   logLik
#> -96.92355 -88.51636 54.46177
#>
#> Random effects:
#> Formula: list(MSS ~ 1, TAU ~ 1)
#> Level: athlete
#> Structure: General positive-definite, Log-Cholesky parametrization
#>      StdDev     Corr
#> MSS      1.32853982 MSS
#> TAU      0.27791928 0.924
#> Residual 0.01519505
#>
#> Fixed effects: MSS + TAU ~ 1
#>      Value Std.Error DF   t-value p-value
#> MSS 8.474621 0.6159646 24 13.758292      0
#> TAU 1.154940 0.1293310 24  8.930112      0
#> Correlation:
#>      MSS
#> TAU 0.923
#>
#> Standardized Within-Group Residuals:
#>      Min        Q1        Med        Q3        Max
#> -1.3358323 -0.4066588 -0.1325837  0.3284514  2.7084751
#>
#> Number of Observations: 30
#> Number of Groups: 5

```

And time_correction can also be used in predict_ and find_ family of functions:

```

velocity_over_distance_corrected <- merge(mixed_model_corrected$parameters$random,
                                           data.frame(distance = seq(0, 40, length.out = 1000)))

velocity_over_distance_corrected$pred_velocity <- with(velocity_over_distance,

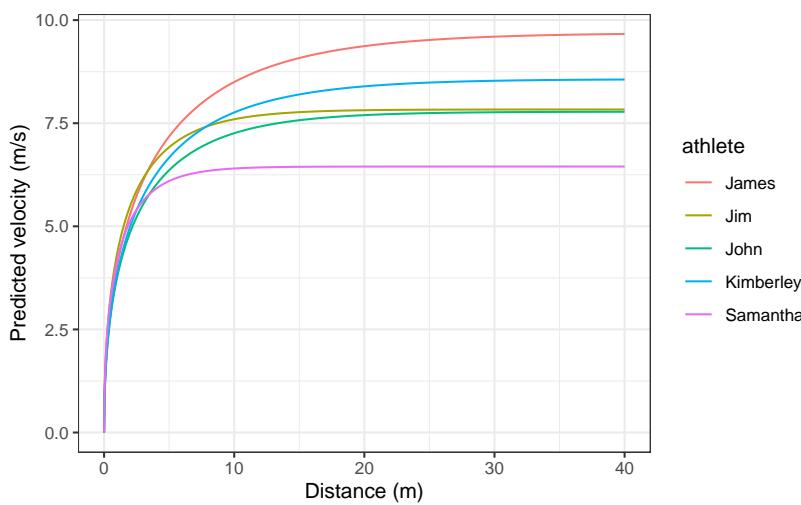
```

```

shorts::predict_velocity_at_distance(distance = distance,
                                     MSS = MSS, TAU = TAU, time_correction = 0.3)

ggplot(velocity_over_distance_corrected, aes(x = distance,
                                              y = pred_velocity, color = athlete)) + theme_bw() + geom_line() +
  xlab("Distance (m)") + ylab("Predicted velocity (m/s)")

```



Instead of providing for `time_correction`, this parameter can be estimated using `shorts::model_using_splits_with_time_correction` and `shorts::mixed_model_using_splits_with_time_correction`:

```

kimberley_profile_with_time_correction <- shorts::model_using_splits_with_time_correction(
  distance = kimberley_data$distance,
  time = kimberley_data$time)

kimberley_profile_with_time_correction
#> Estimated model parameters
#> -----
#>          MSS           TAU
#>     8.9748353      1.2348565
#>          MAC           PMAX
#>     7.2679175      16.3070907
#> time_correction distance_correction
#>     0.2346537      0.0000000
#>
#> Model fit estimators
#> -----
#>      RSE      R_squared      minErr      maxErr

```

```

#> 0.0011290466 0.9999996942 -0.0012094658 0.0011807342
#> maxAbsErr RMSE MAE MAPE
#> 0.0012094658 0.0007983565 0.0006586035 0.0282352643

# Mixed-effect model using `time_correction` as fixed
# effect only To use `time_correction` as random effects,
# use random = MSS + TAU + time_correction ~ 1
mixed_model_with_time_correction <- shorts::mixed_model_using_splits_with_time_correct(
  distance = "distance", time = "time", athlete = "athlete")

# Parameters
mixed_model_with_time_correction
#> Estimated fixed model parameters
#> -----
#>          MSS           TAU
#> 8.3040140        0.9687348
#>          MAC           PMAX
#> 8.5720197        17.7955429
#> time_correction distance_correction
#> 0.1989677        0.0000000
#>
#> Estimated frandom model parameters
#> -----
#>   athlete      MSS      TAU      MAC      PMAX
#> 1 James 10.186327 1.2429367 8.195370 20.87018
#> 2 Jim 7.946099 0.7643674 10.395655 20.65123
#> 3 John 7.996262 1.0488272 7.624003 15.24088
#> 4 Kimberley 8.899472 1.1615147 7.661953 17.04683
#> 5 Samantha 6.491911 0.6260282 10.369998 16.83028
#> time_correction distance_correction
#> 1 0.1989677        0
#> 2 0.1989677        0
#> 3 0.1989677        0
#> 4 0.1989677        0
#> 5 0.1989677        0
#>
#> Model fit estimators
#> -----
#>       RSE    R_squared      minErr      maxErr
#> 0.005976815 0.999990286 -0.016508275 0.009370607
#> maxAbsErr RMSE MAE MAPE
#> 0.016508275 0.004882226 0.003481096 0.186135567

```

For more details, please refer to `sprint-corrections` vignette².

²<https://mladenjovanovic.github.io/shorts/articles/sprint-corrections.html>

18.2.4 Leave-One-Out Cross-Validation (LOOCV)

...`model_using_splits`.. family of functions come with LOOCV feature that is performed by setting the function parameter `LOOCV = TRUE`. This feature is very useful for checking model parameters robustness and model predictions on unseen data. LOOCV involve iterative model building and testing by removing observation one by one and making predictions for them. Let's use Kimberly again, but this time perform LOOCV:

```
kimberley_profile_LOOCV <- shorts::model_using_splits(distance = kimberley_data$distance,
  time = kimberley_data$time, LOOCV = TRUE)

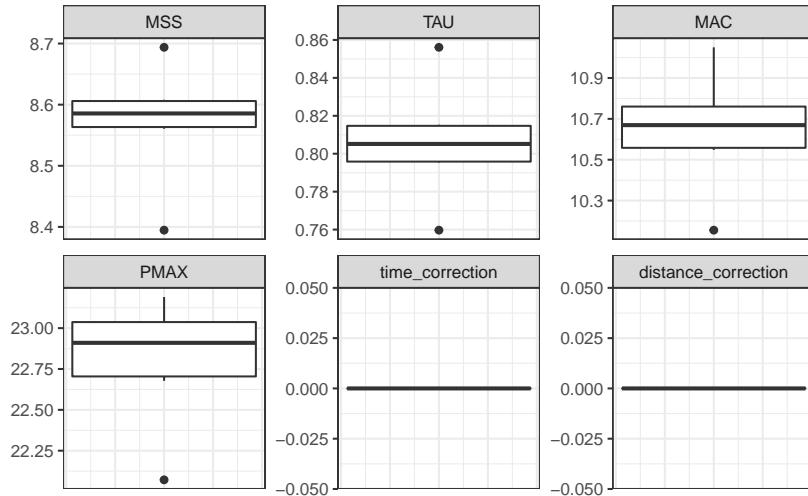
kimberley_profile_LOOCV
#> Estimated model parameters
#> -----
#>          MSS           TAU
#> 8.5911421 0.8113282
#>          MAC           PMAX
#> 10.5889855 22.7428698
#> time_correction distance_correction
#> 0.0000000 0.0000000
#>
#> Model fit estimators
#> -----
#>      RSE   R_squared     minErr     maxErr
#> 0.03403413 0.99965531 -0.02699169 0.05293444
#> maxAbsErr      RMSE       MAE       MAPE
#> 0.05293444 0.02778875 0.02333342 1.19263116
#>
#>
#> Leave-One-Out Cross-Validation
#> -----
#> Parameters:
#>          MSS           TAU           MAC           PMAX time_correction
#> 1 8.693799 0.8561004 10.15512 22.07164 0
#> 2 8.599598 0.8152654 10.54822 22.67761 0
#> 3 8.560665 0.7953644 10.76320 23.03504 0
#> 4 8.571599 0.7972996 10.75079 23.03786 0
#> 5 8.608051 0.8130138 10.58783 22.78514 0
#> 6 8.394673 0.7596923 11.05010 23.19049 0
#> distance_correction
#> 1 0
#> 2 0
#> 3 0
#> 4 0
#> 5 0
```

```
#> 6          0
#>
#> Model fit:
#>   RSE    R_squared      minErr      maxErr
#>   NA  0.99901083 -0.03444984  0.08009048
#>   maxAbsErr      RMSE      MAE      MAPE
#> 0.08009048  0.04742769  0.03923869  1.72270273
```

Box-plot is suitable method for plotting estimated parameters:

```
LOOCV_parameters <- gather(kimberley_profile_LOOCV$LOOCV$parameters) %>%
  mutate(key = factor(key, levels = c("MSS", "TAU", "MAC",
  "PMAX", "time_correction", "distance_correction")))

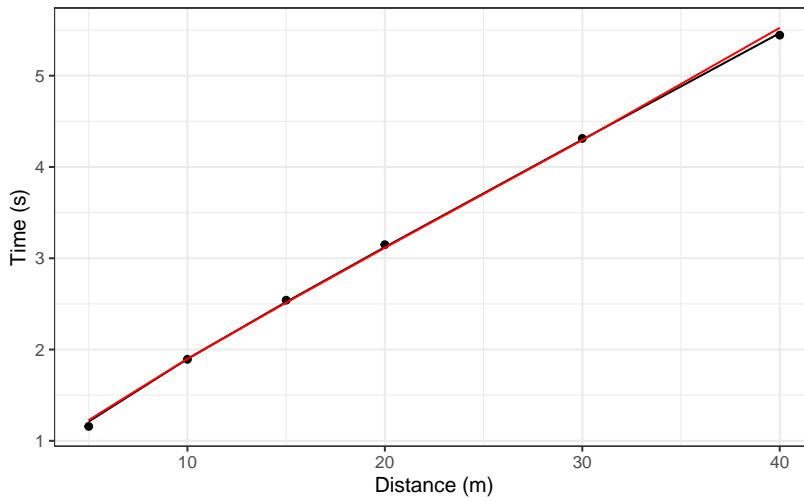
ggplot(LOOCV_parameters, aes(y = value)) + theme_bw() + geom_boxplot() +
  facet_wrap(~key, scales = "free") + ylab(NULL) + theme(axis.ticks.x = element_blank(),
axis.text.x = element_blank())
```



Let's plot model LOOCV predictions and training (when using all data set) predictions against observed performance:

```
kimberley_data <- kimberley_data %>% mutate(pred_time = predict(kimberley_profile_LOOCV$LOOCV_time = kimberley_profile_LOOCV$LOOCV$data$pred_time)

ggplot(kimberley_data, aes(x = distance)) + theme_bw() +
  geom_point(aes(y = time)) + geom_line(aes(y = pred_time),
  color = "black") + geom_line(aes(y = LOOCV_time), color = "red") +
  xlab("Distance (m)") + ylab("Time (s)")
```

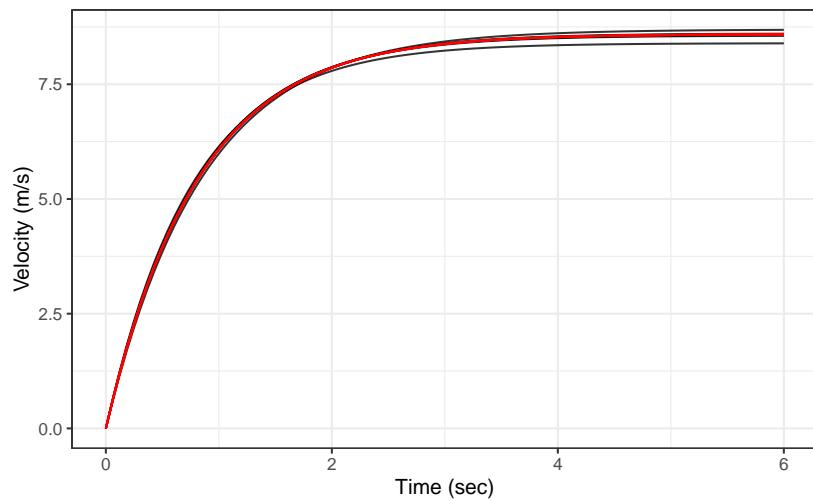


Let's plot predicted velocity using LOOCV estimate parameters to check robustness of the model predictions:

```
plot_data <- kimberley_profile_LOOCV$LOOCV$parameters %>%
  mutate(LOOCV = row_number())

plot_data <- expand_grid(data.frame(time = seq(0, 6, length.out = 100)),
  plot_data) %>% mutate(LOOCV_velocity = predict_velocity_at_time(time = time,
  MSS = MSS, TAU = TAU), velocity = predict_velocity_at_time(time = time,
  MSS = kimberley_profile_LOOCV$parameters$MSS, TAU = kimberley_profile_LOOCV$parameters$TAU))

ggplot(plot_data, aes(x = time, y = LOOCV_velocity, group = LOOCV)) +
  theme_bw() + geom_line(alpha = 0.8) + geom_line(aes(y = velocity),
  color = "red", size = 0.5) + xlab("Time (sec)") + ylab("Velocity (m/s)")
```



18.3 shorts Citation

To cite `shorts`, please use the following command to get the BibTex entry:

Chapter 19

Appendix C: `vjsim` package

`vjsim` (Jovanović 2020b) is R package that simulates vertical jump with the aim of teaching basic biomechanical principles, FV profiling, and exploring assumptions of FV optimization models.

19.1 `vjsim` Installation

You can install the development version from GitHub¹ with:

```
# install.packages("devtools")
devtools::install_github("mladenjovanovic/vjsim")

require(vjsim)
```

19.2 `vjsim` Usage

Please read accompanying vignettes for examples and usage of the `vjsim` package

19.2.1 Introduction to `vjsim`²

This vignette discusses the basic mechanical representation of the vertical jump system. Please read this to understand the overall setup. Access it by clicking the above link or running the following code:

¹<https://github.com/mladenjovanovic/vjsim>

²<https://mladenjovanovic.github.io/vjsim/articles/introduction-vjsim.html>

```
vignette("introduction-vjsim")
```

19.2.2 Simulation³

This vignette continues the Introduction⁴ vignette and expands on the topic of simulation and how vertical jump is simulated. Access it by clicking the above link or running the following code:

```
vignette("simulation-vjsim")
```

19.2.3 Profiling⁵

Once you understand how the Simulation⁶ works, we can start playing with profiling. Force-Velocity (FV), Load-Velocity (LV), and other profiles are discussed. Access it by clicking the above link or running the following code:

```
vignette("profiling-vjsim")
```

19.2.4 Optimization⁷

In this vignette I will introduce few optimization models, influenced by the work of Pierre Samozino and Jean-Benoit Morin. Access it by clicking the above link or running the following code:

```
vignette("optimization-vjsim")
```

19.2.5 Exploring⁸

In this vignette we are going to explore various assumptions of the model, “optimal” FV profiles and some other interesting questions. Access it by clicking the above link or running the following code:

```
vignette("exploring-vjsim")
```

³<https://mladenjovanovic.github.io/vjsim/articles/simulation-vjsim.html>

⁴<https://mladenjovanovic.github.io/vjsim/articles/introduction-vjsim.html>

⁵<https://mladenjovanovic.github.io/vjsim/articles/profiling-vjsim.html>

⁶<https://mladenjovanovic.github.io/vjsim/articles/simulation-vjsim.html>

⁷<https://mladenjovanovic.github.io/vjsim/articles/optimization-vjsim.html>

⁸<https://mladenjovanovic.github.io/vjsim/articles/exploring-vjsim.html>

19.2.6 Modeling⁹

In this vignette I am going to show you how you can use `vjsim` to create athlete profiles from collected data

```
vignette("modeling-vjsim")
```

19.3 Shiny App¹⁰

To run the Shiny app, use the following code, or by clicking on the above link (this will take you to the *shinyapps.io*)

```
# install.packages(c("shiny", "plotly", "DT"))
run_simulator()
```

19.4 vjsim Example

`vjsim` comes with an example data (`testing_data`) that can be used to demonstrate the profiling:

```
require(vjsim)
require(tidyverse)

data("testing_data")

testing_data
#>   athlete bodyweight push_off_distance external_load
#> 1   John     100          0.45            0
#> 2   John     100          0.45           20
#> 3   John     100          0.45           40
#> 4   John     100          0.45           60
#> 5   John     100          0.45           80
#> 6   Jack      85          0.35            0
#> 7   Jack      85          0.35           20
#> 8   Jack      85          0.35           40
#> 9   Jack      85          0.35           60
#> 10  Peter     95          0.50            0
#> 11  Peter     95          0.50           20
#> 12  Peter     95          0.50           40
```

⁹<https://mladenjovanovic.github.io/vjsim/articles/modeling-vjsim.html>

¹⁰<https://athletess.shinyapps.io/shiny-simulator/>

```

#> 13 Peter 95 0.50 60
#> 14 Peter 95 0.50 100
#> 15 Jane 55 0.30 0
#> 16 Jane 55 0.30 10
#> 17 Jane 55 0.30 20
#> 18 Jane 55 0.30 30
#> 19 Jane 55 0.30 40
#> 20 Chris 75 NA 0
#> 21 Chris 75 NA 20
#> 22 Chris 75 NA 40
#> 23 Chris 75 NA 60
#> 24 Chris 75 NA 80
#> aerial_time
#> 1 0.5393402
#> 2 0.4692551
#> 3 0.4383948
#> 4 0.3869457
#> 5 0.3383155
#> 6 0.4991416
#> 7 0.4188160
#> 8 0.3664766
#> 9 0.2746293
#> 10 0.6908913
#> 11 0.5973635
#> 12 0.5546890
#> 13 0.4896103
#> 14 0.4131987
#> 15 0.4589603
#> 16 0.4310090
#> 17 0.3717250
#> 18 0.3354157
#> 19 0.3197858
#> 20 0.4943675
#> 21 0.4618711
#> 22 0.4318388
#> 23 0.4013240
#> 24 0.3696963

```

As can be seen from the above listing, `testing_data` contains data for N=5 athletes. Each athlete performed progressive squat jumps (external load is indicated in `external_load` column; weight are in kg) and the flight time (`aerial_time`) in seconds is measured using the jump mat. Some measuring devices will return *peak velocity* (PV), *take-off velocity* (TOV), or calculated *jump height* (height) which is the most common.

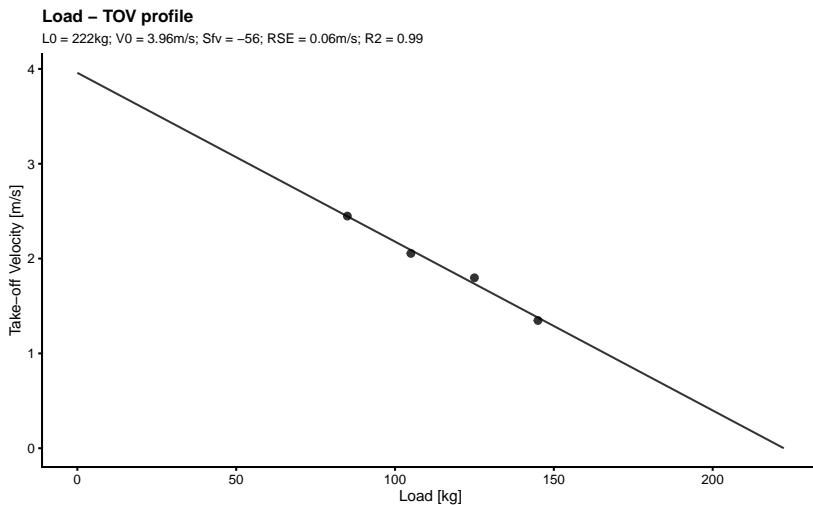
We will select Jack and show his *Load-Velocity* (LV) profile using `vsim` function

```
vjsim::make_load_profile12:

# Filter only Jack's data
jack_data <- filter(testing_data, athlete == "Jack")

# Make a LV profile Here we have used `with` command to
# simplify the code If not we need to use:
# vjsim::make_load_profile( bodyweight =
# jack_data$bodyweight, external_load =
# jack_data$external_load, aerial_time =
# jack_data$aerial_time, plot = TRUE )

jack_LV_profile <- with(jack_data, vjsim::make_load_profile(bodyweight = bodyweight,
    external_load = external_load, aerial_time = aerial_time,
    plot = TRUE))
```



Velocity, in this case TOV, is calculated by first calculating jump height from the `aerial_time` using `vjsim::get_height_from_aerial_time` function inside the `vjsim::make_load_profile`, which is later converted to TOV using ballistic equations.

This LV profile has the following parameters:

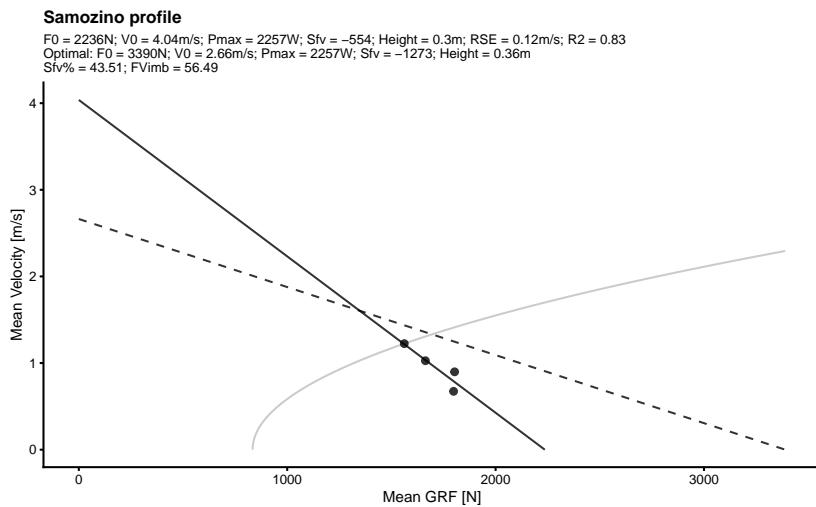
```
jack_LV_profile
#> $LO
#> [1] 222.3918
```

¹²You can omit typing `vjsim::` in the code. I use it here to indicate functions from the `vjsim` package).

```
#>
#> $L0_rel
#> [1] 2.616374
#>
#> $TOVO
#> [1] 3.959045
#>
#> $Sltv
#> [1] -56.17309
#>
#> $Sltv_rel
#> [1] -0.6608599
#>
#> $RSE
#> [1] 0.0560288
#>
#> $R_squared
#> [1] 0.9901916
```

To perform optimization *Force-Velocity* (FV) profile using Samozino *et al.* approach (Jiménez-Reyes et al. 2017; Jiménez-Reyes, Samozino, and Morin 2019; Samozino et al. 2010, 2012, 2008; Samozino 2018c, 2018b, 2018a) use `vjsim::make_samozi_profile`:

```
jack_FV_profile <- with(jack_data, vjsim::make_samozi_profile(bodyweight = bodyweight,
                                                               push_off_distance = push_off_distance, external_load = external_load,
                                                               aerial_time = aerial_time, plot = TRUE))
```



Here are the parameters estimated:

```
jack_FV_profile
#> $F0
#> [1] 2235.995
#>
#> $F0_rel
#> [1] 26.30583
#>
#> $V0
#> [1] 4.037337
#>
#> $Pmax
#> [1] 2256.867
#>
#> $Pmax_rel
#> [1] 26.55137
#>
#> $Sfv
#> [1] -553.8293
#>
#> $Sfu_rel
#> [1] -6.515638
#>
#> $take_off_velocity
#> [1] 2.444065
#>
#> $height
#> [1] 0.3044574
#>
#> $optimal_F0
#> [1] 3389.95
#>
#> $optimal_F0_rel
#> [1] 39.88176
#>
#> $optimal_V0
#> [1] 2.663009
#>
#> $optimal_height
#> [1] 0.3614484
#>
#> $optimal_height_diff
#> [1] 0.05699097
#>
#> $optimal_height_ratio
```

```
#> [1] 1.187189
#>
#> $optimal_Pmax
#> [1] 2256.867
#>
#> $optimal_Pmax_rel
#> [1] 26.55137
#>
#> $optimal_take_off_velocity
#> [1] 2.663009
#>
#> $optimal_take_off_velocity_diff
#> [1] 0.2189439
#>
#> $optimal_take_off_velocity_ratio
#> [1] 1.089582
#>
#> $optimal_Sfv
#> [1] -1272.977
#>
#> $optimal_Sfv_rel
#> [1] -14.9762
#>
#> $Sfv_perc
#> [1] 43.50661
#>
#> $FV_imbalance
#> [1] 56.49339
#>
#> $probe_IMB
#> [1] 42.45125
#>
#> $RSE
#> [1] 0.1172931
#>
#> $R_squared
#> [1] 0.8280585
```

In the case that you need to make analysis for multiple athletes, rather than doing one-by-one, use the following `tidyverse` wrapper:

```
make_samozino_profile_wrapper <- function(data) {
  profile <- with(data, vjsim::make_samozino_profile(bodyweight = bodyweight,
    push_off_distance = push_off_distance, external_load = external_load,
    aerial_time = aerial_time, plot = FALSE))
```

```

return(data.frame(F0 = profile$F0, V0 = profile$V0, height = profile$height,
                 optimal_F0 = profile$optimal_F0, optimal_V0 = profile$optimal_V0,
                 optimal_height = profile$optimal_height, Sfv_perc = profile$Sfv_perc))
}

```

We can now apply this wrapper function to get FV profile for all athletes in the testing_data data set:

```

athlete_profiles <- testing_data %>% group_by(athlete) %>%
  do(make_samozino_profile_wrapper(.))

athlete_profiles
#> # A tibble: 5 x 8
#> # Groups:   athlete [5]
#>   athlete    F0     V0 height optimal_F0 optimal_V0
#>   <fct>  <dbl>  <dbl>  <dbl>      <dbl>      <dbl>
#> 1 John     3373.  2.72  0.351     3495.     2.63
#> 2 Jack     2236.  4.04  0.304     3390.     2.66
#> 3 Peter    3772.  3.49  0.562     3964.     3.32
#> 4 Jane     1961.  2.31  0.259     2011.     2.25
#> 5 Chris     NA    NA     NA        NA        NA
#> # ... with 2 more variables: optimal_height <dbl>,
#> #   Sfv_perc <dbl>

```


Chapter 20

Appendix D: Recommended material

General

1. Dienes Z. 2008. Understanding Psychology as a Science: An Introduction to Scientific and Statistical Inference. New York: Red Globe Press.
2. Ellenberg J, Ellenberg J. 2014. How not to be wrong: the hidden maths of everyday life. New York, New York: Penguin Books.
3. Foreman JW. 2014. Data smart: using data science to transform information into insight. Hoboken, New Jersey: John Wiley & Sons.
4. Gelman A, Hill J, Vehtari A. 2020. Regression and Other Stories. S.l.: Cambridge University Press.
5. Spiegelhalter D. 2019. The art of statistics: how to learn from data. New York: Basic Books, an imprint of Perseus Books, a subsidiary of Hachette Book Group.

Bayesian analysis

7. Kruschke JK. 2015. Doing Bayesian data analysis: a tutorial with R, JAGS, and Stan. Boston: Academic Press.
8. Lambert B. 2018. A student's guide to Bayesian statistics. Los Angeles: SAGE.
9. McElreath R. 2020. Statistical rethinking: a Bayesian course with examples in R and Stan. Boca Raton: Taylor and Francis, CRC Press.

10. Stanton JM. 2017. Reasoning with data: an introduction to traditional and Bayesian statistics using R. New York: The Guilford Press.

Predictive analysis and Machine Learning

11. James G, Witten D, Hastie T, Tibshirani R. 2017. An Introduction to Statistical Learning: with Applications in R. New York: Springer.
12. Kuhn M, Johnson K. 2018. Applied Predictive Modeling. New York: Springer.
13. Kuhn M, Johnson K. 2019. Feature Engineering and Selection: a Practical Approach for Predictive Models. Milton: CRC Press LLC.
14. Lantz B. 2019. Machine learning with R: expert techniques for predictive modeling.
15. Rhys HI. 2020. Machine Learning with R, the tidyverse, and mlr. Manning Publications.

Causal inference

16. Hernán MA, Robins J. 2019. Causal Inference. Boca Raton: Chapman & Hall/CRC.
17. Kleinberg S. 2015. Why: A Guide to Finding and Using Causes. Beijing; Boston: O'Reilly Media.
18. Pearl J, Mackenzie D. 2018. The Book of Why: The New Science of Cause and Effect. New York: Basic Books.

R Language and Visualization

19. Healy K. 2018. Data visualization: a practical introduction. Princeton, NJ: Princeton University Press.
20. Kabacoff R. 2015. R in action: data analysis and graphics with R. Shelter Island: Manning.
21. Matloff NS. 2011. The art of R programming: tour of statistical software design. San Francisco: No Starch Press.
22. Wickham H, Grolemund G. 2016. R for data science: import, tidy, transform, visualize, and model data. Sebastopol, CA: O'Reilly.
23. Wickham H. 2019. Advanced R, Second Edition. Boca Raton: Chapman and Hall/CRC.

24. Wilke C. 2019. Fundamentals of data visualization: a primer on making informative and compelling figures. Sebastopol, CA: O'Reilly Media.

Simulation and statistical inference

25. Carsey T, Harden J. 2013. Monte Carlo Simulation and Resampling Methods for Social Science. Los Angeles: Sage Publications, Inc.
26. Efron B, Hastie T. 2016. Computer Age Statistical Inference: Algorithms, Evidence, and Data Science. New York, NY: Cambridge University Press.

Online Courses

27. Hastie T, Tibshirani R. 2016. Statistical Learning. Available at <https://lagunita.stanford.edu/courses/HumanitiesSciences/StatLearning/Winter2016/about> (accessed October 16, 2019).
28. Lakens D. 2017. Improving your statistical inferences. Available at <https://www.coursera.org/learn/statistical-inferences> (accessed October 16, 2019).
29. Lakens D. 2019. Improving your statistical questions. Available at <https://www.coursera.org/learn/improving-statistical-questions> (accessed October 16, 2019).

Chapter 21

References

- Albanese, Davide, Michele Filosi, Roberto Visintainer, Samantha Riccadonna, Giuseppe Jurman, and Cesare Furlanello. 2012a. “Minerva and Minepy: A c Engine for the Mine Suite and Its R, Python and Matlab Wrappers.” *Bioinformatics*, bts707.
- . 2012b. “Minerva and Minepy: A C Engine for the MINE Suite and Its R, Python and MATLAB Wrappers.” *Bioinformatics*, bts707.
- Allaire, JJ, Jeffrey Horner, Yihui Xie, Vicent Marti, and Natacha Porte. 2019. *Markdown: Render Markdown with the c Library ‘Sundown’*. <https://CRAN.R-project.org/package=markdown>.
- Allen, Mary J., and Wendy M. Yen. 2001. *Introduction to Measurement Theory*. 1 edition. Long Grove, Ill: Waveland Pr Inc.
- Allen, Micah, Davide Poggiali, Kirstie Whitaker, Tom Rhys Marshall, and Rogier Kievit. 2018. “Raincloudplots Tutorials and Codebase.” Zenodo. <https://doi.org/10.5281/zenodo.1402959>.
- Allen, Micah, Davide Poggiali, Kirstie Whitaker, Tom Rhys Marshall, and Rogier A. Kievit. 2019. “Raincloud Plots: A Multi-Platform Tool for Robust Data Visualization.” *Wellcome Open Research* 4 (April): 63. <https://doi.org/10.12688/wellcomeopenres.15191.1>.
- Altmann, Thomas, Jakob Bodensteiner, Cord Dankers, Thommy Dassen, Nikolas Fritz, Sebastian Gruber, Philipp Kopper, Veronika Kronseder, Moritz Wagner, and Emanuel Renkl. 2019. *Limitations of Interpretable Machine Learning Methods*.
- Amrhein, Valentin, David Trafimow, and Sander Greenland. 2019. “Inferential Statistics as Descriptive Statistics: There Is No Replication Crisis If We Don’t Expect Replication.” *The American Statistician* 73 (sup1): 262–70. <https://doi.org/10.1080/00031305.2018.1543137>.

- Angrist, Joshua David, and Jörn-Steffen Pischke. 2015. *Mastering 'Metrics: The Path from Cause to Effect*. Princeton ; Oxford: Princeton University Press.
- Anvari, Farid, and Daniel Lakens. 2019. “Using Anchor-Based Methods to Determine the Smallest Effect Size of Interest,” March. <https://doi.org/10.31234/osf.io/syp5a>.
- Barker, Richard J., and Matthew R. Schofield. 2008. “Inference About Magnitudes of Effects.” *International Journal of Sports Physiology and Performance* 3 (4): 547–57. <https://doi.org/10.1123/ijsspp.3.4.547>.
- Barron, Jonathan T. 2019. “A General and Adaptive Robust Loss Function.” *arXiv:1701.03077 [Cs, Stat]*, April. <http://arxiv.org/abs/1701.03077>.
- Batterham, Alan M., and William G. Hopkins. 2006. “Making Meaningful Inferences About Magnitudes.” *International Journal of Sports Physiology and Performance* 1 (1): 50–57. <https://doi.org/10.1123/ijsspp.1.1.50>.
- Beaujean, A. Alexander. 2014. *Latent Variable Modeling Using R: A Step by Step Guide*. New York: Routledge/Taylor & Francis Group.
- Biecek, Przemyslaw, and Tomasz Burzykowski. 2019. *Predictive Models: Explore, Explain, and Debug*.
- Binmore, Ken. 2011. *Rational Decisions*. Fourth Impression edition. Princeton, NJ: Princeton University Press.
- Bischl, Bernd, Michel Lang, Lars Kotthoff, Julia Schiffner, Jakob Richter, Erich Studerus, Giuseppe Casalicchio, and Zachary M. Jones. 2016. “mlr: Machine Learning in R.” *Journal of Machine Learning Research* 17 (170): 1–5. <http://jmlr.org/papers/v17/15-066.html>.
- Bischl, Bernd, Michel Lang, Jakob Richter, Jakob Bossek, Daniel Horn, and Pascal Kerschke. 2020. *ParamHelpers: Helpers for Parameters in Black-Box Optimization, Tuning and Machine Learning*. <https://CRAN.R-project.org/package=ParamHelpers>.
- Bischl, Bernd, Jakob Richter, Jakob Bossek, Daniel Horn, Janek Thomas, and Michel Lang. 2017. “MlrMBO: A Modular Framework for Model-Based Optimization of Expensive Black-Box Functions.” *arXiv Preprint arXiv:1703.03373*.
- Bland, J. M., and D. G. Altman. 1986. “Statistical Methods for Assessing Agreement Between Two Methods of Clinical Measurement.” *Lancet (London, England)* 1 (8476): 307–10.
- Borg, David N., Geoffrey M. Minett, Ian B. Stewart, and Christopher C. Drovandi. 2018. “Bayesian Methods Might Solve the Problems with Magnitude-Based Inference.” *Medicine & Science in Sports & Exercise* 50 (12): 2609–10. <https://doi.org/10.1249/MSS.0000000000001736>.
- Borsboom, Denny. 2009. *Measuring the Mind: Conceptual Issues in Modern Psychometrics*. Cambridge: Cambridge University Press.

- . 2008. “Latent Variable Theory.” *Measurement: Interdisciplinary Research & Perspective* 6 (1-2): 25–53. <https://doi.org/10.1080/15366360802035497>.
- Borsboom, Denny, Gideon J. Mellenbergh, and Jaap van Heerden. 2003. “The Theoretical Status of Latent Variables.” *Psychological Review* 110 (2): 203–19. <https://doi.org/10.1037/0033-295X.110.2.203>.
- Botchkarev, Alexei. 2019. “A New Typology Design of Performance Metrics to Measure Errors in Machine Learning Regression Algorithms.” *Interdisciplinary Journal of Information, Knowledge, and Management* 14: 045–076. <https://doi.org/10.28945/4184>.
- Breheny, Patrick, and Woodrow Burchett. 2017. “Visualization of Regression Models Using Visreg.” *The R Journal* 9 (2): 56–71.
- Breiman, Leo. 2001. “Statistical Modeling: The Two Cultures.” *Statistical Science* 16 (3): 199–215.
- Buchheit, Martin, and Alireza Rabbani. 2014. “The 2015 Intermittent Fitness Test Versus the Yo-Yo Intermittent Recovery Test Level 1: Relationship and Sensitivity to Training.” *International Journal of Sports Physiology and Performance* 9 (3): 522–24. <https://doi.org/10.1123/ijsspp.2012-0335>.
- Caldwell, Aaron R., and Samuel N. Cheuvront. 2019. “Basic Statistical Considerations for Physiology: The Journal Temperature Toolbox.” *Temperature*, June, 1–30. <https://doi.org/10.1080/23328940.2019.1624131>.
- Canty, Angelo, and B. D. Ripley. 2017. *Boot: Bootstrap R (S-Plus) Functions*.
- Carsey, Thomas, and Jeffrey Harden. 2013. *Monte Carlo Simulation and Resampling Methods for Social Science*. 1 edition. Los Angeles: Sage Publications, Inc.
- Casalicchio, Giuseppe, Jakob Bossek, Michel Lang, Dominik Kirchhoff, Pascal Kerschke, Benjamin Hofner, Heidi Seibold, Joaquin Vanschoren, and Bernd Bischl. 2017. “OpenML: An R Package to Connect to the Machine Learning Platform Openml.” *Computational Statistics*, 1–15.
- Chai, T., and R. R. Draxler. 2014. “Root Mean Square Error (RMSE) or Mean Absolute Error (MAE)? Arguments Against Avoiding RMSE in the Literature.” *Geoscientific Model Development* 7 (3): 1247–50. <https://doi.org/10.5194/gmd-7-1247-2014>.
- Clarke, David C., and Philip F. Skiba. 2013. “Rationale and Resources for Teaching the Mathematical Modeling of Athletic Training and Performance.” *Advances in Physiology Education* 37 (2): 134–52. <https://doi.org/10.1152/advan.00078.2011>.
- Cohen, Jacob. 1988. *Statistical Power Analysis for the Behavioral Sciences*. 2nd ed. Hillsdale, N.J: L. Erlbaum Associates.
- Cumming, Geoff. 2014. “The New Statistics: Why and How.” *Psychological Science* 25 (1): 7–29. <https://doi.org/10.1177/0956797613504966>.

- Curran-Everett, Douglas. 2018. “Magnitude-Based Inference: Good Idea but Flawed Approach.” *Medicine & Science in Sports & Exercise* 50 (10): 2164–5. <https://doi.org/10.1249/MSS.00000000000001646>.
- Dankel, Scott J., and Jeremy P. Loenneke. 2019. “A Method to Stop Analyzing Random Error and Start Analyzing Differential Responders to Exercise.” *Sports Medicine*, June. <https://doi.org/10.1007/s40279-019-01147-0>.
- Davison, A. C., and D. V. Hinkley. 1997a. *Bootstrap Methods and Their Applications*. Cambridge: Cambridge University Press. <http://statwww.epfl.ch/davison/BMA/>.
- . 1997b. *Bootstrap Methods and Their Applications*. Cambridge: Cambridge University Press.
- Dienes, Zoltan. 2008. *Understanding Psychology as a Science: An Introduction to Scientific and Statistical Inference*. 2008 edition. New York: Red Globe Press.
- Efron, Bradley. 2005. “Bayesians, Frequentists, and Scientists.” *Journal of the American Statistical Association* 100 (469): 1–5. <https://doi.org/10.1198/016214505000000033>.
- Efron, Bradley, and Trevor Hastie. 2016. *Computer Age Statistical Inference: Algorithms, Evidence, and Data Science*. 1 edition. New York, NY: Cambridge University Press.
- Estrada, Eduardo, Emilio Ferrer, and Antonio Pardo. 2019. “Statistics for Evaluating Pre-Post Change: Relation Between Change in the Distribution Center and Change in the Individual Scores.” *Frontiers in Psychology* 9 (January). <https://doi.org/10.3389/fpsyg.2018.02696>.
- Everitt, Brian, and Torsten Hothorn. 2011. *An Introduction to Applied Multivariate Analysis with R*. Use R! New York: Springer.
- Finch, W. Holmes, and Brian F. French. 2015. *Latent Variable Modeling with R*. New York: Routledge, Taylor & Francis Group.
- Fisher, Aaron J., John D. Medaglia, and Bertus F. Jeronimus. 2018. “Lack of Group-to-Individual Generalizability Is a Threat to Human Subjects Research.” *Proceedings of the National Academy of Sciences* 115 (27): E6106–E6115. <https://doi.org/10.1073/pnas.1711978115>.
- Foreman, John W. 2014. *Data Smart: Using Data Science to Transform Information into Insight*. Hoboken, New Jersey: John Wiley & Sons.
- Fox, John. 2003. “Effect Displays in R for Generalised Linear Models.” *Journal of Statistical Software* 8 (15): 1–27. <http://www.jstatsoft.org/v08/i15/>.
- Fox, John, and Jangman Hong. 2009. “Effect Displays in R for Multinomial and Proportional-Odds Logit Models: Extensions to the effects Package.” *Journal of Statistical Software* 32 (1): 1–24. <http://www.jstatsoft.org/v32/i01/>.

- Fox, John, and Sanford Weisberg. 2018. “Visualizing Fit and Lack of Fit in Complex Regression Models with Predictor Effect Plots and Partial Residuals.” *Journal of Statistical Software* 87 (9): 1–27. <https://doi.org/10.18637/jss.v087.i09>.
- Fox, John, Sanford Weisberg, and Brad Price. 2019. *CarData: Companion to Applied Regression Data Sets*. <https://CRAN.R-project.org/package=carData>.
- Friedman, Jerome, Trevor Hastie, and Robert Tibshirani. 2010. “Regularization Paths for Generalized Linear Models via Coordinate Descent.” *Journal of Statistical Software* 33 (1): 1–22.
- Gelman, Andrew. 2011. “Causality and Statistical Learning.” *American Journal of Sociology* 117 (3): 955–66. <https://doi.org/10.1086/662659>.
- Gelman, Andrew, and Sander Greenland. 2019. “Are Confidence Intervals Better Termed ‘Uncertainty Intervals’?” *BMJ*, September, l5381. <https://doi.org/10.1136/bmj.l5381>.
- Gelman, Andrew, and Christian Hennig. 2017. “Beyond Subjective and Objective in Statistics.” *Journal of the Royal Statistical Society: Series A (Statistics in Society)* 180 (4): 967–1033. <https://doi.org/10.1111/rssa.12276>.
- Giavarina, Davide. 2015. “Understanding Bland Altman Analysis.” *Biochimia Medica* 25 (2): 141–51. <https://doi.org/10.11613/BM.2015.015>.
- Gigerenzer, Gerd, Ralph Hertwig, and Thorsten Pachur. 2015. *Heuristics: The Foundations of Adaptive Behavior*. Reprint edition. Oxford University Press.
- Glazier, Paul S., and Sina Mehdizadeh. 2018. “Challenging Conventional Paradigms in Applied Sports Biomechanics Research.” *Sports Medicine*, December. <https://doi.org/10.1007/s40279-018-1030-1>.
- Goldstein, Alex, Adam Kapelner, Justin Bleich, and Emil Pitkin. 2013. “Peeking Inside the Black Box: Visualizing Statistical Learning with Plots of Individual Conditional Expectation.” *arXiv:1309.6392 [Stat]*, September. <http://arxiv.org/abs/1309.6392>.
- Greenwell, Brandon, Brad Boehmke, and Bernie Gray. 2020. *Vip: Variable Importance Plots*. <https://CRAN.R-project.org/package=vip>.
- Greenwell, Brandon M. 2017a. “Pdp: An R Package for Constructing Partial Dependence Plots.” *The R Journal* 9 (1): 421–36. <https://journal.r-project.org/archive/2017/RJ-2017-016/index.html>.
- . 2017b. “Pdp: An R Package for Constructing Partial Dependence Plots.” *The R Journal* 9 (1): 421–36. <https://doi.org/10.32614/RJ-2017-016>.
- Hamner, Ben, and Michael Frasco. 2018. *Metrics: Evaluation Metrics for Machine Learning*. <https://CRAN.R-project.org/package=Metrics>.

- Hastie, Trevor, Robert Tibshirani, and J. H. Friedman. 2009. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. 2nd ed. Springer Series in Statistics. New York, NY: Springer.
- Heckman, James J. 2005. “Rejoinder: Response to Sobel.” *Sociological Methodology* 35 (1): 135–50. <https://doi.org/10.1111/j.0081-1750.2006.00166.x>.
- Hecksteden, Anne, Jochen Kraushaar, Friederike Scharhag-Rosenberger, Daniel Theisen, Stephen Senn, and Tim Meyer. 2015. “Individual Response to Exercise Training - a Statistical Perspective.” *Journal of Applied Physiology* 118 (12): 1450–9. <https://doi.org/10.1152/japplphysiol.00714.2014>.
- Hecksteden, Anne, Werner Pitsch, Friederike Rosenberger, and Tim Meyer. 2018. “Repeated Testing for the Assessment of Individual Response to Exercise Training.” *Journal of Applied Physiology* 124 (6): 1567–79. <https://doi.org/10.1152/japplphysiol.00896.2017>.
- Henry, Lionel, and Hadley Wickham. 2020. *Purrr: Functional Programming Tools*. <https://CRAN.R-project.org/package=purrr>.
- Henry, Lionel, Hadley Wickham, and Winston Chang. 2020. *Ggstance: Horizontal 'Ggplot2' Components*. <https://CRAN.R-project.org/package=ggstance>.
- Hernan, M. A. 2002. “Causal Knowledge as a Prerequisite for Confounding Evaluation: An Application to Birth Defects Epidemiology.” *American Journal of Epidemiology* 155 (2): 176–84. <https://doi.org/10.1093/aje/155.2.176>.
- Hernan, M. A., and S. R. Cole. 2009. “Invited Commentary: Causal Diagrams and Measurement Bias.” *American Journal of Epidemiology* 170 (8): 959–62. <https://doi.org/10.1093/aje/kwp293>.
- Hernán, M A, and S L Taubman. 2008. “Does Obesity Shorten Life? The Importance of Well-Defined Interventions to Answer Causal Questions.” *International Journal of Obesity* 32 (S3): S8–S14. <https://doi.org/10.1038/ijo.2008.82>.
- Hernán, Miguel A. 2016. “Does Water Kill? A Call for Less Casual Causal Inferences.” *Annals of Epidemiology* 26 (10): 674–80. <https://doi.org/10.1016/j.annepidem.2016.08.016>.
- . 2017. “Causal Diagrams: Draw Your Assumptions Before Your Conclusions Course | PH559x | edX.” *edX*. <https://courses.edx.org/courses/course-v1:HarvardX+PH559x+3T2017/course/>.
- . 2018. “The C-Word: Scientific Euphemisms Do Not Improve Causal Inference from Observational Data.” *American Journal of Public Health* 108 (5): 616–19. <https://doi.org/10.2105/AJPH.2018.304337>.
- Hernán, Miguel A., John Hsu, and Brian Healy. 2019. “A Second Chance to Get Causal Inference Right: A Classification of Data Science Tasks.” *CHANCE* 32 (1): 42–49. <https://doi.org/10.1080/09332480.2019.1579578>.

- Hernán, Miguel A., and JM Robins. n.d. *Causal Inference*. Boca Raton: Chapman & Hall/CRC.
- Hesterberg, Tim C. 2015. “What Teachers Should Know About the Bootstrap: Resampling in the Undergraduate Statistics Curriculum.” *The American Statistician* 69 (4): 371–86. <https://doi.org/10.1080/00031305.2015.1089789>.
- Hocking, Toby Dylan. 2020. *Directlabels: Direct Labels for Multicolor Plots*. <https://CRAN.R-project.org/package=directlabels>.
- Hopkins, Will. 2015. “Spreadsheets for Analysis of Validity and Reliability.” *Sportscience.org*, 9.
- Hopkins, Will, and Alan Batterham. 2018. “The Vindication of Magnitude-Based Inference,” 12.
- Hopkins, Will G. 2004a. “How to Interpret Changes in an Athletic Performance Test,” November, 2.
- . 2006. “New View of Statistics: Effect Magnitudes.” <https://www.sportsci.org/resource/stats/effectmag.html>.
- Hopkins, Will G. 2015. “Individual Responses Made Easy.” *Journal of Applied Physiology* 118 (12): 1444–6. <https://doi.org/10.1152/japplphysiol.00098.2015>.
- . 2000. “Measures of Reliability in Sports Medicine and Science.” *Sports Med*, 15.
- . 2004b. “Bias in Bland-Altman but Not Regression Validity Analyses.” *Sportscience.org*. <https://sportsci.org/jour/04/wghbias.htm>.
- . 2007. “Understanding Statistics by Using Spreadsheets to Generate and Analyze Samples.” *Sportscience.org*. <https://www.sportsci.org/2007/wghstats.htm>.
- . 2010. “A Socratic Dialogue on Comparison of Measures.” *Sportscience.org*. <http://www.sportsci.org/2010/wghmeasures.htm>.
- Hopkins, William G., Stephen W. Marshall, Alan M. Batterham, and Juri Hanin. 2009. “Progressive Statistics for Studies in Sports Medicine and Exercise Science.” *Medicine & Science in Sports & Exercise* 41 (1): 3–13. <https://doi.org/10.1249/MSS.0b013e31818cb278>.
- J, Twisk, Bosman L, Hoekstra T, Rijnhart J, Welten M, and Heymans M. 2018. “Different Ways to Estimate Treatment Effects in Randomised Controlled Trials.” *Contemporary Clinical Trials Communications* 10 (June): 80–85. <https://doi.org/10.1016/j.conctc.2018.03.008>.
- James, Gareth, Daniela Witten, Trevor Hastie, and Robert Tibshirani. 2017. *An Introduction to Statistical Learning: With Applications in R*. 1st ed. 2013, Corr. 7th printing 2017 edition. New York: Springer.
- Jiménez-Reyes, Pedro, Pierre Samozino, Matt Brughelli, and Jean-Benoît Morin. 2017. “Effectiveness of an Individualized Training Based on Force-Velocity

- Profiling During Jumping.” *Frontiers in Physiology* 7 (January). <https://doi.org/10.3389/fphys.2016.00677>.
- Jiménez-Reyes, Pedro, Pierre Samozino, and Jean-Benoît Morin. 2019. “Optimized Training for Jumping Performance Using the Force-Velocity Imbalance: Individual Adaptation Kinetics.” Edited by Daniel Boullosa. *PLOS ONE* 14 (5): e0216681. <https://doi.org/10.1371/journal.pone.0216681>.
- Jovanovic, Mladen. 2020. “shorts: Short Sprints.” <https://doi.org/10.5281/zenodo.3751836>.
- Jovanovic, Mladen, and Benedict Stephens Hemingway. 2020. “dorem: Dose Response Modeling.” <https://doi.org/10.5281/zenodo.3757085>.
- Jovanović, Mladen. 2020a. *bmbstats: Bootstrap Magnitude-Based Statistics*. Belgrade, Serbia. <https://github.com/mladenjovanovic/bmbstats>.
- . 2020b. “vjsim: Vertical Jump Simulator.” <https://doi.org/10.5281/zenodo.3740291>.
- Jovanović, Mladen. 2020. “Extending the Classical Test Theory with Circular Performance Model.” *Complementary Training*.
- Kabacoff, Robert. 2015. *R in Action: Data Analysis and Graphics with R*. Second edition. Shelter Island: Manning.
- Keogh, Ruth H., Pamela A. Shaw, Paul Gustafson, Raymond J. Carroll, Veronika Deffner, Kevin W. Dodd, Helmut Küchenhoff, et al. 2020. “STRATOS Guidance Document on Measurement Error and Misclassification of Variables in Observational Epidemiology: Part 1-Basic Theory and Simple Methods of Adjustment.” *Statistics in Medicine*, April. <https://doi.org/10.1002/sim.8532>.
- King, Madeleine T. 2011. “A Point of Minimal Important Difference (MID): A Critique of Terminology and Methods.” *Expert Review of Pharmacoeconomics & Outcomes Research* 11 (2): 171–84. <https://doi.org/10.1586/erp.11.9>.
- Kleinberg, Jon, Annie Liang, and Sendhil Mullainathan. 2017. “The Theory Is Predictive, but Is It Complete? An Application to Human Perception of Randomness.” *arXiv:1706.06974 [Cs, Stat]*, June. <http://arxiv.org/abs/1706.06974>.
- Kleinberg, Samantha. 2018. *Causality, Probability, and Time*.
- . 2015. *Why: A Guide to Finding and Using Causes*. 1 edition. Beijing ; Boston: O'Reilly Media.
- Kruschke, John K. 2013. “Bayesian Estimation Supersedes the T Test.” *Journal of Experimental Psychology: General* 142 (2): 573–603. <https://doi.org/10.1037/a0029146>.
- Kruschke, John K., and Torrin M. Liddell. 2018a. “Bayesian Data Analysis for Newcomers.” *Psychonomic Bulletin & Review* 25 (1): 155–77. <https://doi.org/10.3758/s13423-017-1272-1>.

- . 2018b. “The Bayesian New Statistics: Hypothesis Testing, Estimation, Meta-Analysis, and Power Analysis from a Bayesian Perspective.” *Psychonomic Bulletin & Review* 25 (1): 178–206. <https://doi.org/10.3758/s13423-016-1221-4>.
- Kuhn, Max. 2020. *Caret: Classification and Regression Training*. <https://CRAN.R-project.org/package=caret>.
- Kuhn, Max, and Kjell Johnson. 2019. *Feature Engineering and Selection: A Practical Approach for Predictive Models*. Milton: CRC Press LLC.
- . 2018. *Applied Predictive Modeling*. 1st ed. 2013, Corr. 2nd printing 2016 edition. New York: Springer.
- Kuhn, Max, Jed Wing, Steve Weston, Andre Williams, Chris Keefer, Allan Engelhardt, Tony Cooper, et al. 2018. *Caret: Classification and Regression Training*.
- Lakens, Daniël. 2017. “Equivalence Tests: A Practical Primer for *T* Tests, Correlations, and Meta-Analyses.” *Social Psychological and Personality Science* 8 (4): 355–62. <https://doi.org/10.1177/1948550617697177>.
- Lakens, Daniël, Anne M. Scheel, and Peder M. Isager. 2018. “Equivalence Testing for Psychological Research: A Tutorial.” *Advances in Methods and Practices in Psychological Science* 1 (2): 259–69. <https://doi.org/10.1177/2515245918770963>.
- Lang, Kyle M., Shauna J. Sweet, and Elizabeth M. Grandfield. 2017. “Getting Beyond the Null: Statistical Modeling as an Alternative Framework for Inference in Developmental Science.” *Research in Human Development* 14 (4): 287–304. <https://doi.org/10.1080/15427609.2017.1371567>.
- Lang, Michel, Martin Binder, Jakob Richter, Patrick Schratz, Florian Pfisterer, Stefan Coors, Quay Au, Giuseppe Casalicchio, Lars Kotthoff, and Bernd Bischl. 2019. “mlr3: A Modern Object-Oriented Machine Learning Framework in R.” *Journal of Open Source Software*, December. <https://doi.org/10.21105/joss.01903>.
- Lang, Michel, Helena Kotthaus, Peter Marwedel, Claus Weihs, Joerg Rahnenfuehrer, and Bernd Bischl. 2014. “Automatic Model Selection for High-Dimensional Survival Analysis.” *Journal of Statistical Computation and Simulation* 85 (1): 62–76.
- Lantz, Brett. 2019. *Machine Learning with R: Expert Techniques for Predictive Modeling*.
- Lederer, David J., Scott C. Bell, Richard D. Branson, James D. Chalmers, Rachel Marshall, David M. Maslove, David E. Ost, et al. 2019. “Control of Confounding and Reporting of Results in Causal Inference Studies. Guidance for Authors from Editors of Respiratory, Sleep, and Critical Care Journals.” *Annals of the American Thoracic Society* 16 (1): 22–28. <https://doi.org/10.1513/AnnalsATS.201808-564PS>.

- Lederer, Wolfgang, and Helmut Küchenhoff. 2006. “A Short Introduction to the SIMEX and MCSIMEX.” *R News* 6 (January).
- Ludbrook, John. 1997. “SPECIAL ARTICLE COMPARING METHODS OF MEASUREMENT.” *Clinical and Experimental Pharmacology and Physiology* 24 (2): 193–203. <https://doi.org/10.1111/j.1440-1681.1997.tb01807.x>.
- . 2002. “Statistical Techniques for Comparing Measurers and Methods of Measurement: A Critical Review.” *Clinical and Experimental Pharmacology and Physiology* 29 (7): 527–36. <https://doi.org/10.1046/j.1440-1681.2002.03686.x>.
- . 2010. “Linear Regression Analysis for Comparing Two Measurers or Methods of Measurement: But Which Regression?: Linear Regression for Comparing Methods.” *Clinical and Experimental Pharmacology and Physiology* 37 (7): 692–99. <https://doi.org/10.1111/j.1440-1681.2010.05376.x>.
- . 2012. “A Primer for Biomedical Scientists on How to Execute Model II Linear Regression Analysis: Model II Linear Regression Analysis.” *Clinical and Experimental Pharmacology and Physiology* 39 (4): 329–35. <https://doi.org/10.1111/j.1440-1681.2011.05643.x>.
- Lübke, Karsten, Matthias Gehrke, Jörg Horst, and Gero Szepannek. 2020. “Why We Should Teach Causal Inference: Examples in Linear Regression with Simulated Data.” *Journal of Statistics Education*, May, 1–7. <https://doi.org/10.1080/10691898.2020.1752859>.
- Makowski, Dominique, Mattan Ben-Shachar, and Daniel Lüdecke. 2019a. “bayestestR: Describing Effects and Their Uncertainty, Existence and Significance Within the Bayesian Framework.” *Journal of Open Source Software* 4 (40): 1541. <https://doi.org/10.21105/joss.01541>.
- Makowski, Dominique, Mattan S. Ben-Shachar, SH Annabel Chen, and Daniel Lüdecke. n.d. “Indices of Effect Existence and Significance in the Bayesian Framework.” <https://doi.org/10.31234/osf.io/2zexr>.
- Makowski, Dominique, Mattan S. Ben-Shachar, and Daniel Lüdecke. 2019. “BayestestR: Describing Effects and Their Uncertainty, Existence and Significance Within the Bayesian Framework.” *Journal of Open Source Software* 4 (40): 1541. <https://doi.org/10.21105/joss.01541>.
- Makowski, Dominique, Mattan S. Ben-Shachar, and Daniel Lüdecke. 2019b. “Understand and Describe Bayesian Models and Posterior Distributions Using bayestestR.” CRAN. <https://doi.org/10.5281/zenodo.2556486>.
- McElreath, Richard. 2015. *Statistical Rethinking: A Bayesian Course with Examples in R and Stan*. 1 edition. Boca Raton: Chapman and Hall/CRC.
- McGraw, Kenneth O., and S. P. Wong. 1992. “A Common Language Effect Size Statistic.” *Psychological Bulletin* 111 (2): 361–65. <https://doi.org/10.1037/0033-2909.111.2.361>.

- Miller, Tim. 2017. "Explanation in Artificial Intelligence: Insights from the Social Sciences." *arXiv:1706.07269 [Cs]*, June. <http://arxiv.org/abs/1706.07269>.
- Mitchell, Sandra. 2012. *Unsimple Truths: Science, Complexity, and Policy*. Paperback ed. Chicago, Mich.: The Univ. of Chicago Press.
- Mitchell, Sandra D. 2002. "Integrative Pluralism." *Biology & Philosophy* 17 (1): 55–70. <https://doi.org/10.1023/A:1012990030867>.
- Molenaar, Peter C. M. 2004. "A Manifesto on Psychology as Idiographic Science: Bringing the Person Back into Scientific Psychology, This Time Forever." *Measurement: Interdisciplinary Research & Perspective* 2 (4): 201–18. https://doi.org/10.1207/s15366359mea0204_1.
- Molenaar, Peter C. M., and Cynthia G. Campbell. 2009. "The New Person-Specific Paradigm in Psychology." *Current Directions in Psychological Science* 18 (2): 112–17. <https://doi.org/10.1111/j.1467-8721.2009.01619.x>.
- Molnar, Christoph. 2018. *Interpretable Machine Learning*. Leanpub.
- Molnar, Christoph, Bernd Bischl, and Giuseppe Casalicchio. 2018. "Iml: An R Package for Interpretable Machine Learning." *JOSS* 3 (26): 786. <https://doi.org/10.21105/joss.00786>.
- Morey, Richard D., Rink Hoekstra, Jeffrey N. Rouder, Michael D. Lee, and Eric-Jan Wagenmakers. 2016. "The Fallacy of Placing Confidence in Confidence Intervals." *Psychonomic Bulletin & Review* 23 (1): 103–23. <https://doi.org/10.3758/s13423-015-0947-8>.
- Mullineaux, David R., Christopher A. Barnes, and Alan M. Batterham. 1999. "Assessment of Bias in Comparing Measurements: A Reliability Example." *Measurement in Physical Education and Exercise Science* 3 (4): 195–205. https://doi.org/10.1207/s15327841mpee0304_1.
- Müller, Kirill, and Hadley Wickham. 2020. *Tibble: Simple Data Frames*. <https://CRAN.R-project.org/package=tibble>.
- Nevill, Alan M., A. Mark Williams, Colin Boreham, Eric S. Wallace, Gareth W. Davison, Grant Abt, Andrew M. Lane, and Edward M. Winter EDITORIAL BOARD. 2018. "Can We Trust 'Magnitude-Based Inference'?" *Journal of Sports Sciences* 36 (24): 2769–70. <https://doi.org/10.1080/02640414.2018.1516004>.
- Norman, Geoffrey R., Femida Gwadry Sridhar, Gordon H. Guyatt, and Stephen D. Walter. 2001. "Relation of Distribution- and Anchor-Based Approaches in Interpretation of Changes in Health-Related Quality of Life:" *Medical Care* 39 (10): 1039–47. <https://doi.org/10.1097/00005650-200110000-00002>.
- Novick, Melvin R. 1966. "The Axioms and Principal Results of Classical Test Theory." *Journal of Mathematical Psychology* 3 (1): 1–18. [https://doi.org/10.1016/0022-2496\(66\)90002-2](https://doi.org/10.1016/0022-2496(66)90002-2).

- O'Hagan, Tony. 2004. "Dicing with the Unknown." *Significance* 1 (3): 132–33. <https://doi.org/10.1111/j.1740-9713.2004.00050.x>.
- Page, Scott E. 2018. *The Model Thinker: What You Need to Know to Make Data Work for You*. Basic Books.
- Pearl, Judea. 2009. "Causal Inference in Statistics: An Overview." *Statistics Surveys* 3 (0): 96–146. <https://doi.org/10.1214/09-SS057>.
- . 2019. "The Seven Tools of Causal Inference, with Reflections on Machine Learning." *Communications of the ACM* 62 (3): 54–60. <https://doi.org/10.1145/3241036>.
- Pearl, Judea, Madelyn Glymour, and Nicholas P. Jewell. 2016. *Causal Inference in Statistics: A Primer*. 1 edition. Chichester, West Sussex: Wiley.
- Pearl, Judea, and Dana Mackenzie. 2018. *The Book of Why: The New Science of Cause and Effect*. 1 edition. New York: Basic Books.
- Pinheiro, Jose, Douglas Bates, Saikat DebRoy, Deepayan Sarkar, and R Core Team. 2020. "nlme: Linear and Nonlinear Mixed Effects Models." <https://CRAN.R-project.org/package=nlme>.
- Probst, Philipp, Quay Au, Giuseppe Casalicchio, Clemens Stachl, and Bernd Bischl. 2017. "Multilabel Classification with R Package Mlr." *arXiv Preprint arXiv:1703.08991*.
- R Core Team. 2018. *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing.
- . 2020. *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing. <https://www.R-project.org/>.
- Reshef, D. N., Y. A. Reshef, H. K. Finucane, S. R. Grossman, G. McVean, P. J. Turnbaugh, E. S. Lander, M. Mitzenmacher, and P. C. Sabeti. 2011. "Detecting Novel Associations in Large Data Sets." *Science* 334 (6062): 1518–24. <https://doi.org/10.1126/science.1205438>.
- Revelle, William. 2019. *Psych: Procedures for Psychological, Psychometric, and Personality Research*. Evanston, Illinois: Northwestern University. <https://CRAN.R-project.org/package=psych>.
- Ribeiro, Marco Tulio, Sameer Singh, and Carlos Guestrin. 2016. "'Why Should I Trust You?': Explaining the Predictions of Any Classifier." *arXiv:1602.04938 [Cs, Stat]*, February. <http://arxiv.org/abs/1602.04938>.
- Rohrer, Julia M. 2018. "Thinking Clearly About Correlations and Causation: Graphical Causal Models for Observational Data." *Advances in Methods and Practices in Psychological Science* 1 (1): 27–42. <https://doi.org/10.1177/2515245917745629>.

- Rousselet, Guillaume A., Cyril R. Pernet, and Rand R. Wilcox. 2017. "Beyond Differences in Means: Robust Graphical Methods to Compare Two Groups in Neuroscience." *European Journal of Neuroscience* 46 (2): 1738–48. <https://doi.org/10.1111/ejn.13610>.
- Rousselet, Guillaume A., Cyril R Pernet, and Rand R. Wilcox. 2019a. "A Practical Introduction to the Bootstrap: A Versatile Method to Make Inferences by Using Data-Driven Simulations." <https://doi.org/10.31234/osf.io/h8ft7>.
- . 2019b. "The Percentile Bootstrap: A Teaser with Step-by-Step Instructions in R." <https://doi.org/10.31234/osf.io/kxarf>.
- RStudio Team. 2016. *RStudio: Integrated Development Environment for R*. Boston, MA: RStudio, Inc.
- Saddiki, Hachem, and Laura B. Balzer. 2018. "A Primer on Causality in Data Science." *arXiv:1809.02408 [Stat]*, September. <http://arxiv.org/abs/1809.02408>.
- Sainani, Kristin L. 2012. "Clinical Versus Statistical Significance." *PM&R* 4 (6): 442–45. <https://doi.org/10.1016/j.pmrj.2012.04.014>.
- . 2018. "The Problem with "Magnitude-Based Inference"?" *Medicine and Science in Sports and Exercise* 50 (10): 2166–76. <https://doi.org/10.1249/MSS.0000000000001645>.
- Sainani, Kristin L., Keith R. Lohse, Paul Remy Jones, and Andrew Vickers. 2019. "Magnitude-Based Inference Is Not Bayesian and Is Not a Valid Method of Inference." *Scandinavian Journal of Medicine & Science in Sports*, May. <https://doi.org/10.1111/sms.13491>.
- Samozino, Pierre. 2018a. "A Simple Method for Measuring Force, Velocity and Power Capabilities and Mechanical Effectiveness During Sprint Running." In *Biomechanics of Training and Testing*, edited by Jean-Benoit Morin and Pierre Samozino, 237–67. Cham: Springer International Publishing. https://doi.org/10.1007/978-3-319-05633-3_11.
- . 2018b. "A Simple Method for Measuring Lower Limb Force, Velocity and Power Capabilities During Jumping." In *Biomechanics of Training and Testing*, edited by Jean-Benoit Morin and Pierre Samozino, 65–96. Cham: Springer International Publishing. https://doi.org/10.1007/978-3-319-05633-3_4.
- . 2018c. "Optimal Force-Velocity Profile in Ballistic Push-Off: Measurement and Relationship with Performance." In *Biomechanics of Training and Testing*, edited by Jean-Benoit Morin and Pierre Samozino, 97–119. Cham: Springer International Publishing. https://doi.org/10.1007/978-3-319-05633-3_5.
- Samozino, Pierre, Jean-Benoît Morin, Frédérique Hintzy, and Alain Belli. 2008. "A Simple Method for Measuring Force, Velocity and Power Output During

- Squat Jump.” *Journal of Biomechanics* 41 (14): 2940–5. <https://doi.org/10.1016/j.jbiomech.2008.07.028>.
- . 2010. “Jumping Ability: A Theoretical Integrative Approach.” *Journal of Theoretical Biology* 264 (1): 11–18. <https://doi.org/10.1016/j.jtbi.2010.01.021>.
- Samozino, Pierre, Enrico Rejc, Pietro Enrico Di Prampero, Alain Belli, and Jean-Benoît Morin. 2012. “Optimal ForceVelocity Profile in Ballistic MovementsAltius:” *Medicine & Science in Sports & Exercise* 44 (2): 313–22. <https://doi.org/10.1249/MSS.0b013e31822d757a>.
- Sarkar, Deepayan. 2008. *Lattice: Multivariate Data Visualization with R*. New York: Springer. <http://lmdvr.r-forge.r-project.org>.
- Savage, Leonard J. 1972. *The Foundations of Statistics*. 2nd Revised ed. edition. New York: Dover Publications.
- Shang, Yi. 2012. “Measurement Error Adjustment Using the SIMEX Method: An Application to Student Growth Percentiles: *Measurement Error Adjustment Using the SIMEX Method*.” *Journal of Educational Measurement* 49 (4): 446–65. <https://doi.org/10.1111/j.1745-3984.2012.00186.x>.
- Shaw, Pamela A., Paul Gustafson, Raymond J. Carroll, Veronika Deffner, Kevin W. Dodd, Ruth H. Keogh, Victor Kipnis, et al. 2020. “STRATOS Guidance Document on Measurement Error and Misclassification of Variables in Observational Epidemiology: Part 2-More Complex Methods of Adjustment and Advanced Topics.” *Statistics in Medicine*, April. <https://doi.org/10.1002/sim.8531>.
- Shmueli, Galit. 2010. “To Explain or to Predict?” *Statistical Science* 25 (3): 289–310. <https://doi.org/10.1214/10-STS330>.
- Shrier, Ian, and Robert W Platt. 2008. “Reducing Bias Through Directed Acyclic Graphs.” *BMC Medical Research Methodology* 8 (1). <https://doi.org/10.1186/1471-2288-8-70>.
- Swinton, Paul A., Ben Stephens Hemingway, Bryan Saunders, Bruno Gualano, and Eimear Dolan. 2018. “A Statistical Framework to Interpret Individual Response to Intervention: Paving the Way for Personalized Nutrition and Exercise Prescription.” *Frontiers in Nutrition* 5 (May). <https://doi.org/10.3389/fnut.2018.00041>.
- Tenan, Matthew, Andrew David Vigotsky, and Aaron R Caldwell. n.d. “On the Statistical Properties of the Dankel-Loenneke Method.” <https://doi.org/10.31236/osf.io/8ndhg>.
- Textor, Johannes, Benito van der Zander, Mark S. Gilthorpe, Maciej Liśkiewicz, and George T. H. Ellison. 2017. “Robust Causal Inference Using Directed Acyclic Graphs: The R Package ‘Dagitty’.” *International Journal of Epidemiology*, January, dyw341. <https://doi.org/10.1093/ije/dyw341>.

- Therneau, Terry, and Beth Atkinson. 2019. *Rpart: Recursive Partitioning and Regression Trees*. <https://CRAN.R-project.org/package=rpart>.
- Turner, Anthony, Jon Brazier, Chris Bishop, Shyam Chavda, Jon Cree, and Paul Read. 2015. “Data Analysis for Strength and Conditioning Coaches: Using Excel to Analyze Reliability, Differences, and Relationships.” *Strength and Conditioning Journal* 37 (1): 76–83. <https://doi.org/10.1519/SSC.0000000000000113>.
- Vaughan, Davis, and Max Kuhn. 2020. *Hardhat: Construct Modeling Packages*. <https://CRAN.R-project.org/package=hardhat>.
- Wagenmakers, Eric-Jan. 2007. “A Practical Solution to the Pervasive Problems Ofp Values.” *Psychonomic Bulletin & Review* 14 (5): 779–804. <https://doi.org/10.3758/BF03194105>.
- Wallace, Michael. 2020. “Analysis in an Imperfect World.” *Significance* 17 (1): 14–19. <https://doi.org/10.1111/j.1740-9713.2020.01353.x>.
- Watts, Duncan J, Emorie D Beck, Elisa Jayne Bienenstock, Jake Bowers, Aaron Frank, Anthony Grubasic, Jake Hofman, Julia Marie Rohrer, and Matthew Salganik. 2018. “Explanation, Prediction, and Causality: Three Sides of the Same Coin?” October. <https://doi.org/10.31219/osf.io/u6vz5>.
- Weinberg, Gabriel, and Lauren McCann. 2019. *Super Thinking: The Big Book of Mental Models*. New York: Portfolio/Penguin.
- Welsh, Alan H., and Emma J. Knight. 2015. “‘Magnitude-Based Inference’: A Statistical Review.” *Medicine & Science in Sports & Exercise* 47 (4): 874–84. <https://doi.org/10.1249/MSS.000000000000451>.
- Wickham, Hadley. 2016. *Ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York. <https://ggplot2.tidyverse.org>.
- . 2019. *Stringr: Simple, Consistent Wrappers for Common String Operations*. <https://CRAN.R-project.org/package=stringr>.
- . 2020. *Forcats: Tools for Working with Categorical Variables (Factors)*. <https://CRAN.R-project.org/package=forcats>.
- Wickham, Hadley, Mara Averick, Jennifer Bryan, Winston Chang, Lucy D’Agostino McGowan, Romain François, Garrett Grolemund, et al. 2019. “Welcome to the tidyverse.” *Journal of Open Source Software* 4 (43): 1686. <https://doi.org/10.21105/joss.01686>.
- Wickham, Hadley, Romain François, Lionel Henry, and Kirill Müller. 2020. *Dplyr: A Grammar of Data Manipulation*. <https://CRAN.R-project.org/package=dplyr>.
- Wickham, Hadley, and Lionel Henry. 2020. *Tidyr: Tidy Messy Data*. <https://CRAN.R-project.org/package=tidyr>.

- Wickham, Hadley, Jim Hester, and Romain Francois. 2018. *Readr: Read Rectangular Text Data*. <https://CRAN.R-project.org/package=readr>.
- Wikipedia contributors. 2019. “Causal Model.”
- Wilcox, Rand, Travis J. Peterson, and Jill L. McNitt-Gray. 2018. “Data Analyses When Sample Sizes Are Small: Modern Advances for Dealing with Outliers, Skewed Distributions, and Heteroscedasticity.” *Journal of Applied Biomechanics* 34 (4): 258–61. <https://doi.org/10.1123/jab.2017-0269>.
- Wilcox, Rand R. 2016. *Introduction to Robust Estimation and Hypothesis Testing*. 4th edition. Waltham, MA: Elsevier.
- Wilcox, Rand R., and Guillaume A. Rousselet. 2017. “A Guide to Robust Statistical Methods in Neuroscience.” *bioRxiv*, June. <https://doi.org/10.1101/151811>.
- Wilke, Claus O. 2019. *Cowplot: Streamlined Plot Theme and Plot Annotations for 'Ggplot2'*. <https://CRAN.R-project.org/package=cowplot>.
- . 2020. *Ggridges: Ridgeline Plots in 'Ggplot2'*. <https://CRAN.R-project.org/package=ggridges>.
- Willmott, Cj, and K Matsuura. 2005. “Advantages of the Mean Absolute Error (MAE) over the Root Mean Square Error (RMSE) in Assessing Average Model Performance.” *Climate Research* 30: 79–82. <https://doi.org/10.3354/cr030079>.
- Xie, Yihui. 2015. *Dynamic Documents with R and Knitr*. 2nd ed. Boca Raton, Florida: Chapman; Hall/CRC. <https://yihui.org/knitr/>.
- . 2016. *Bookdown: Authoring Books and Technical Documents with R Markdown*. Boca Raton, Florida: Chapman; Hall/CRC. <https://github.com/rstudio/bookdown>.
- Yarkoni, Tal, and Jacob Westfall. 2017. “Choosing Prediction over Explanation in Psychology: Lessons from Machine Learning.” *Perspectives on Psychological Science* 12 (6): 1100–1122. <https://doi.org/10.1177/1745691617693393>.
- Zhao, Qingyuan, and Trevor Hastie. 2019. “Causal Interpretations of Black-Box Models.” *Journal of Business & Economic Statistics*, July, 1–10. <https://doi.org/10.1080/07350015.2019.1624293>.
- Zhu, Hao. 2019. *KableExtra: Construct Complex Table with 'Kable' and Pipe Syntax*. <https://CRAN.R-project.org/package=kableExtra>.