Statistical Methods to Reduce the Effects of Measurement Error in Sport and Exercise: A Guide for Practitioners and Applied Researchers

Paul A. Swinton[1], Ben Stephens Hemingway[1], Iain J Gallagher[2], Eimear Dolan[3]

Doi:
SportRχiv hosted preprint version 1
24/01/2023
PREPRINT - NOT PEER REVIEWED

1. School of Health Sciences, Robert Gordon University, Aberdeen, UK.

2. School of Applied Sciences, Edinburgh Napier University, Edinburgh, UK

3. Applied Physiology & Nutrition Research Group, Faculdade de Medicina FMUSP, Universidade de Sao Paulo, Sao Paulo, Brazil.

Corresponding Author
Dr. Paul Swinton
School of Health Sciences, Robert Gordon University
Garthdee Road
Aberdeen, UK,
AB10 7QG
p.swinton@rgu.ac.uk, +44 (0) 1224 262 3361

Please cite as: Swinton, PA. Stephens Hemingway B, Gallagher IJ, Dolan E. Statistical methods to reduce the effects of measurement error in sport and exercise: A guide for practitioners and applied researchers. Pre-print available from SportRχiv. https://doi.org10.

ABSTRACT

Quantifying uncertainty in measurements is essential to inform, monitor and evaluate interventions in sport and exercise. Many commonly used tests, particularly those that measure maximum performance or fitness exhibit large measurement errors creating uncertainty that complicates interpretations and decision making. Uncertainty in measurements can be especially problematic where expected changes across an intervention are relatively small. The purpose of the present review is to describe statistical approaches to reduce uncertainty in measurements and therein improve interpretation and decision making. These approaches include increased data collection and the use of relatively simple calculations including means and linear regression to reduce uncertainty. The review provides detailed information on the assumptions underlying each approach and the relevant statistical properties. Visuals and worked examples including R code are provided to solidify concepts and better enable practitioners and applied researchers to adopt the approaches.

1. INTRODUCTION

Practitioners and applied researchers in sport and exercise routinely select and deliver interventions for individuals and groups with the intention of improving one or more physical attributes. The suitability of measurements to inform, monitor and evaluate interventions is dependent on obtaining plausible estimates of the stable state of the underlying attributes. Providing an accurate estimate of a physical attribute is not, however, as simple as taking the numerical score observed via a standard test at face value. These so-called "observed scores" generally comprise some instrumentation and biological noise, collectively known as measurement error (Swinton *et al*, 2018). As a result, practitioners and applied researchers should consider a range of approaches to appropriately quantify measurement error in observed test scores, and where possible, reduce this error and subsequently reflect on the uncertainty when selecting, monitoring and evaluating interventions.

Several approaches exist to reduce the magnitude of errors within observed measurements. For example, standardisation of set-up, calibration, and robust testing protocols may reduce instrumentation noise. Similarly, standardisation of external factors (e.g. time of testing, nutritional intake, and activity performed prior to testing) may reduce error due to biological noise (Swinton *et al*, 2018). These approaches, however, are unlikely to eradicate measurement error completely. Whilst there are clear statistical approaches to conceptualise and ultimately estimate uncertainty in observed scores (Swinton *et al*, 2018; Hopkins, 2000; Hopkins, 2004), often in sport and exercise the methods used create such large measurement errors that they are of limited practical use. Previously (Swinton *et al*, 2018), we described a statistical framework to investigate response to an intervention that focussed on: 1) establishing baseline scores and the degree of measurement error around these scores; and 2) establishing whether or not meaningful change occurred across an intervention using change scores and confidence intervals (CI's). The purpose of the present review is to describe extensions to this framework, in order to enhance its practical application. Throughout the article, we will briefly recap on core concepts previously described, before expanding with methods to reduce measurement error. We encourage interested readers to familiarise themselves with the previous review to better follow the information described herein (Swinton *et al*, 2018). The extensions described will be contextualised with example scenarios and data. The approaches outlined are pragmatic in nature, employing relatively simple statistical models such that most calculations can be incorporated into standard spreadsheets and all can be run in R. Throughout the review a number of practical examples with mock data will be followed with R code available in the supplementary files.

2. ESTABLISHING PLAUSIBLE BASELINE SCORES

2.1 Classical test theory, observed score assumptions, and measurement error.

Initial measurements (baseline scores) can establish the necessity and scope of an intervention prior to time and monetary investment. Additionally, without accurate initial reference points, meaningful statements regarding the occurrence or magnitude of change across an intervention cannot be made. Quantifying measurement error is required to appropriately interpret baseline information. Briefly, an individual's true score (T_s) in a test is regarded as their current stable level, and is always inaccessible, as measurements incorporate some error (ϵ). Therefore, any measurement from a test is referred to as an observed score (O_s) and is considered an estimate of the true score, with $O_s = T_s + \epsilon$. A primary assumption under classical test theory is that if it were possible to conduct a very large number of non-interacting tests on the same individual, then the values observed would follow a normal (Gaussian) distribution with mean equal to the true score, and standard deviation (σ) describing dispersion around the mean. This measurement generating process is therefore described by the distribution $O_s \sim N(T_s, \sigma^2)$.

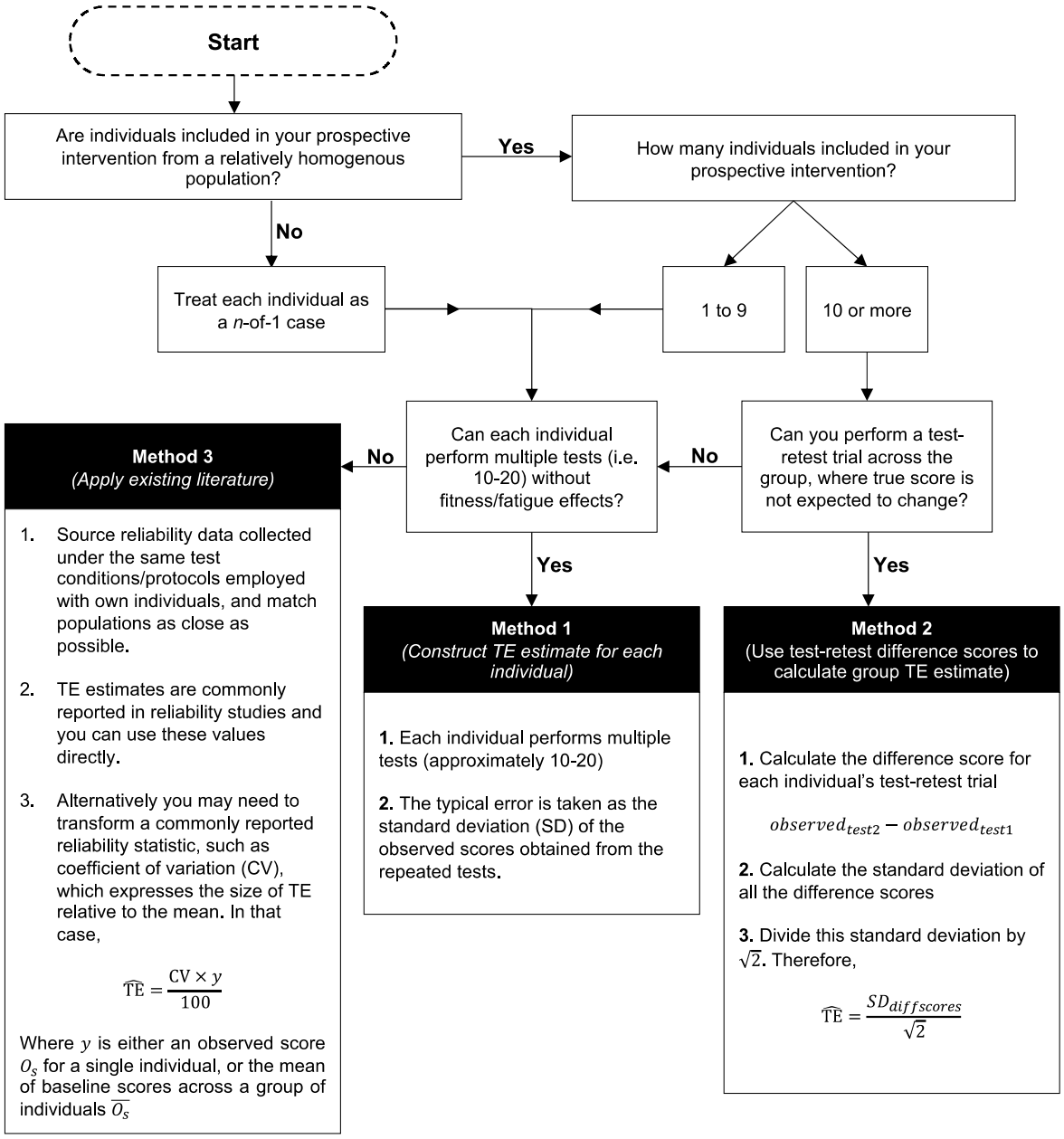
Tests that frequently produce large measurement errors increase the likelihood that observed scores will be inaccurate such that conclusions drawn and interventions adopted, may be unnecessary, ineffective or indeed inappropriate. Under the framework proposed in Swinton *et al,* (2018), measurement error is separated into two primary sources including instrumentation noise and biological noise. Instrumentation noise is error in observed scores caused solely by the set-up, mechanisms, or use of measurement apparatus. Biological noise is error in observed scores created by biological processes, including, but not limited to, phenomena such as circadian rhythm, nutritional intake, sleep and motivation (Hopkins, 2000).

2.2 Methods of calculating typical error

Typical error describes the variation in observed scores caused by measurement error when an individual performs repeated tests (Hopkins, 2000). Typical error is an estimate of the standard deviation (σ) around the true score (T_s), assuming that observed scores are normally distributed for an individual or sample from a population. We typically denote typical error by the notation $\widehat{\mathrm{TE}}$, to reinforce that it is an estimate of the standard deviation describing the spread of scores centred on the true score. There are three primary methods for obtaining a value for typical error in practice: (1) calculating the standard deviation of multiple repeated tests performed by a single individual; (2) using

test-retest difference scores from a moderate to large group over time periods where true scores are not expected to change; (3) using reliability statistics presented in previous research conducted on a sample from a matching population. Each of these methods is briefly summarised in Figure 1, and readers are referred to section 1.1. of Swinton *et al,* (2018) for a more in-depth discussion of the typical error estimate for each method. Regardless of the method implemented, each observed score collected at baseline is considered as a draw from a sampling distribution with standard deviation approximately equal to the typical error estimate obtained.

Figure 1: Decision diagram for selecting the appropriate typical error estimation ($\widehat{\text{TE}}$) method.

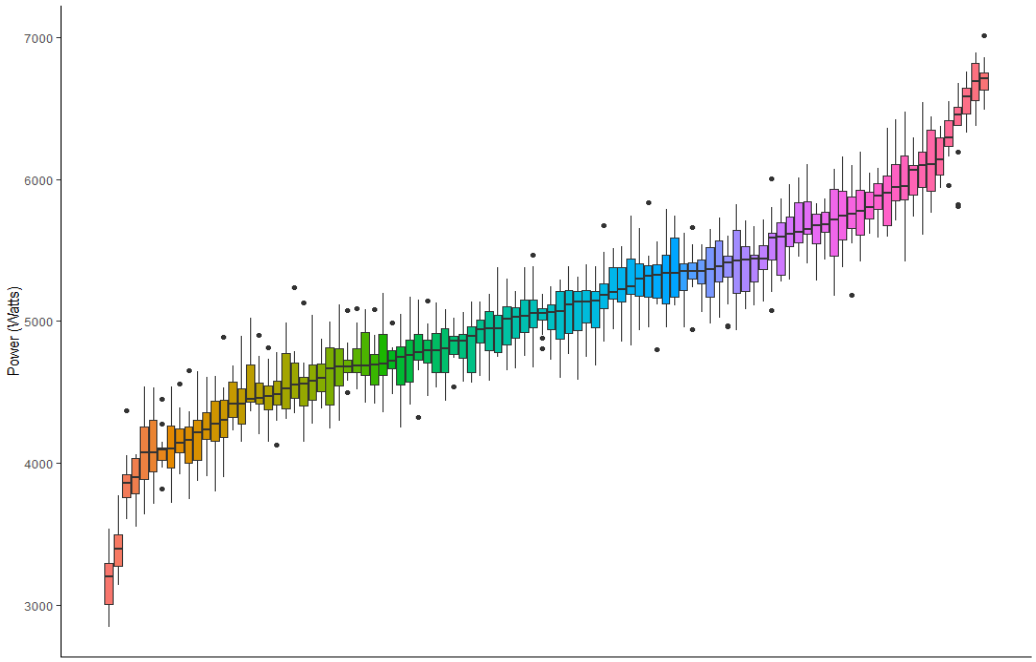


2.3 Constructing true score confidence intervals

Once an observed score(s) has been obtained and an appropriate procedure used to calculate typical error of the test, a CI can be constructed. CI's around observed scores quantify uncertainty and provide a plausible range within which the true score may lie. Given the approximation of the true score as the mean of a large number of independent tests and observed scores following a normal distribution, error in any single measurement is equally likely to be positive or negative. As a result, true score CI's are built by adding and subtracting a multiple of the typical error to each observed score (i.e. $O_s \pm M \cdot \widehat{TE}$). The multiple (M) selected is dependent on the width of the CI desired (for example, 68% or 95%) and the likely precision of $\widehat{TE}$. CI's are a property of a procedure, and when used repeatedly, the percentage of intervals that include the true value will match the CI used. Individual CI's should not be interpreted probabilistically, as it is possible the true score may lie substantially outside the bounds calculated.

To highlight the process of constructing true score intervals we now introduce the mock data that will be used throughout this review. The data comprise measurements of peak mechanical power measured in Watts collected during a vertical jump test. Data are generated for 100 individuals over a pre intervention period of two weeks, and across an intervention phase of 12 weeks. The R code and explanations of the data generation process are presented in the supplementary files. Figure 2 illustrates the baseline data comprising daily measurement for the two week period, with the widths and symmetric nature of boxplots reflecting the assumptions of normally distributed data with measurement error consistent across participants.

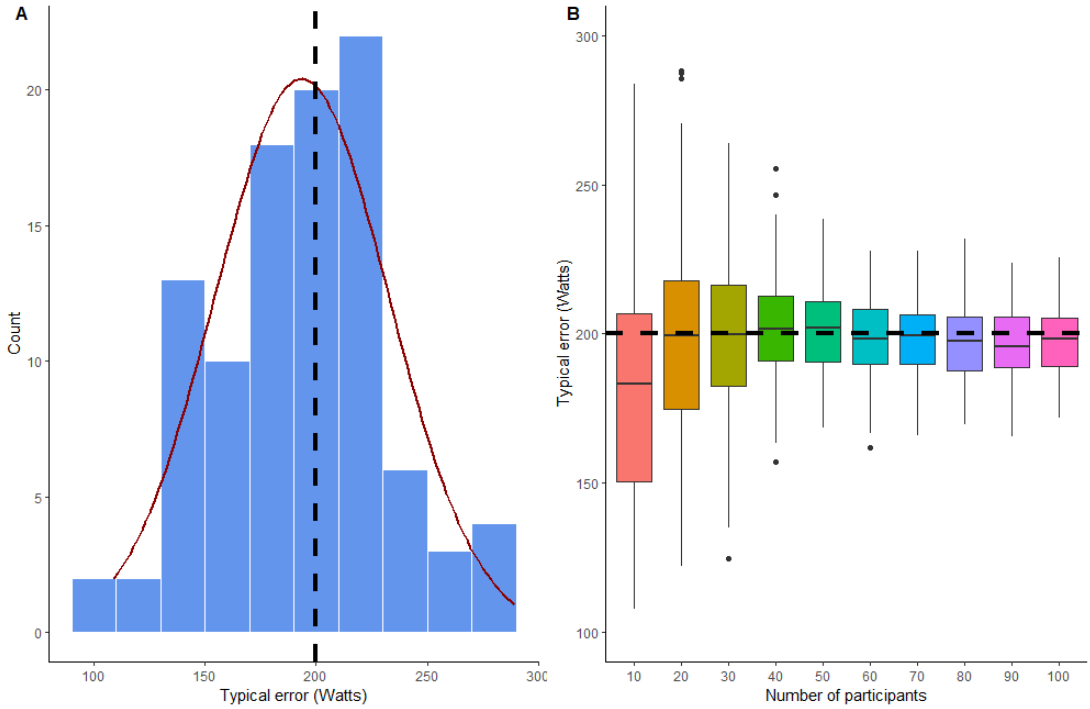
Figure 2: Illustration of mock baseline power data generated for 100 participants



True scores are simulated from $N(3000, 750^2)$, with 14 observed scores obtained for each participant with measurement error distributed as $N(0, 200^2)$. Each box-plot illustrates the distribution of the observed scores from a single participant, with data ordered based on the median value.

We can use this baseline data to calculate $\widehat{TE}$ from either individual participant standard deviations, or from group test-retest. Using the standard deviation of the fourteen data points across the different participants, we can see (Figure 3A) the middle of the distribution is close to the actual measurement error standard deviation (σ = 200 Watts), with the sample median [IQR] equal to 193 [166 to 223 Watts]). Even with fourteen data points, however, calculated $\widehat{TE}$ can be limited (range: 109 to 289). Using the single test-retest method, we can see that $\widehat{TE}$ improves and distributions become tighter around the actual measurement error with increasing sample size (Figure 3B).

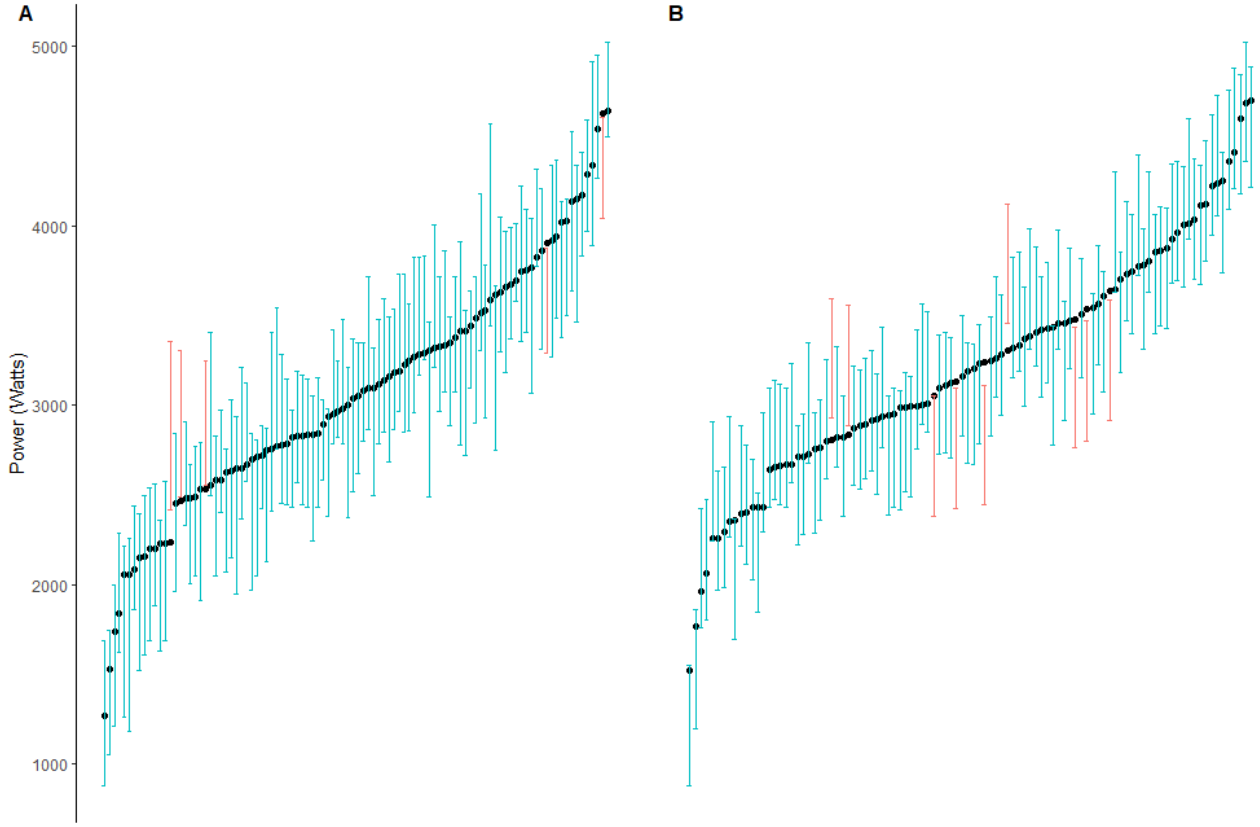
Figure 3: Visualisations of typical error calculations using standard deviation of individual participant data (A) and test-retest data (B).



Plot A: Histogram with superimposed normal curve showing sample distribution of typical errors calculated from participant standard deviation. Plot B: Boxplot showing sample distribution of typical errors calculated from test-retest data from sample sizes ranging from 10 to 100 participants.

For typical error calculated with both individual participant standard deviation and sample test-retest, we see that when calculating true score CI's the proportion that contain the true scores match the corresponding percentage. Figure 4A and 4B illustrate this process, but also highlight how wide intervals are when we have appreciable measurement error. If we wish to obtain 95% true score CI's with for example a test-retest $\widehat{TE}$ from 20 individuals, then our intervals using the data generated here (σ = 200 Watts) will typically equal $O_s \pm$ 420 Watts. This value is obtained using the multiplier M=2.1 (see supplementary files). Given some participants in the population have true scores ~1500 Watts, we can see that this process may be of limited practical use and therefore approaches to reduce uncertainty are warranted.

Figure 4: Visualisations of true score confidence intervals calculated using standard deviation of individual participant data (A) and test-retest data (B).



Circles represent true scores from 100 participants sampled from the population. Intervals represent 95% true score confidence intervals. Intervals are centred on observed scores and those that do not include the true score are highlighted in red. Note that intervals on test-retest data in plot B are all of same magnitude.

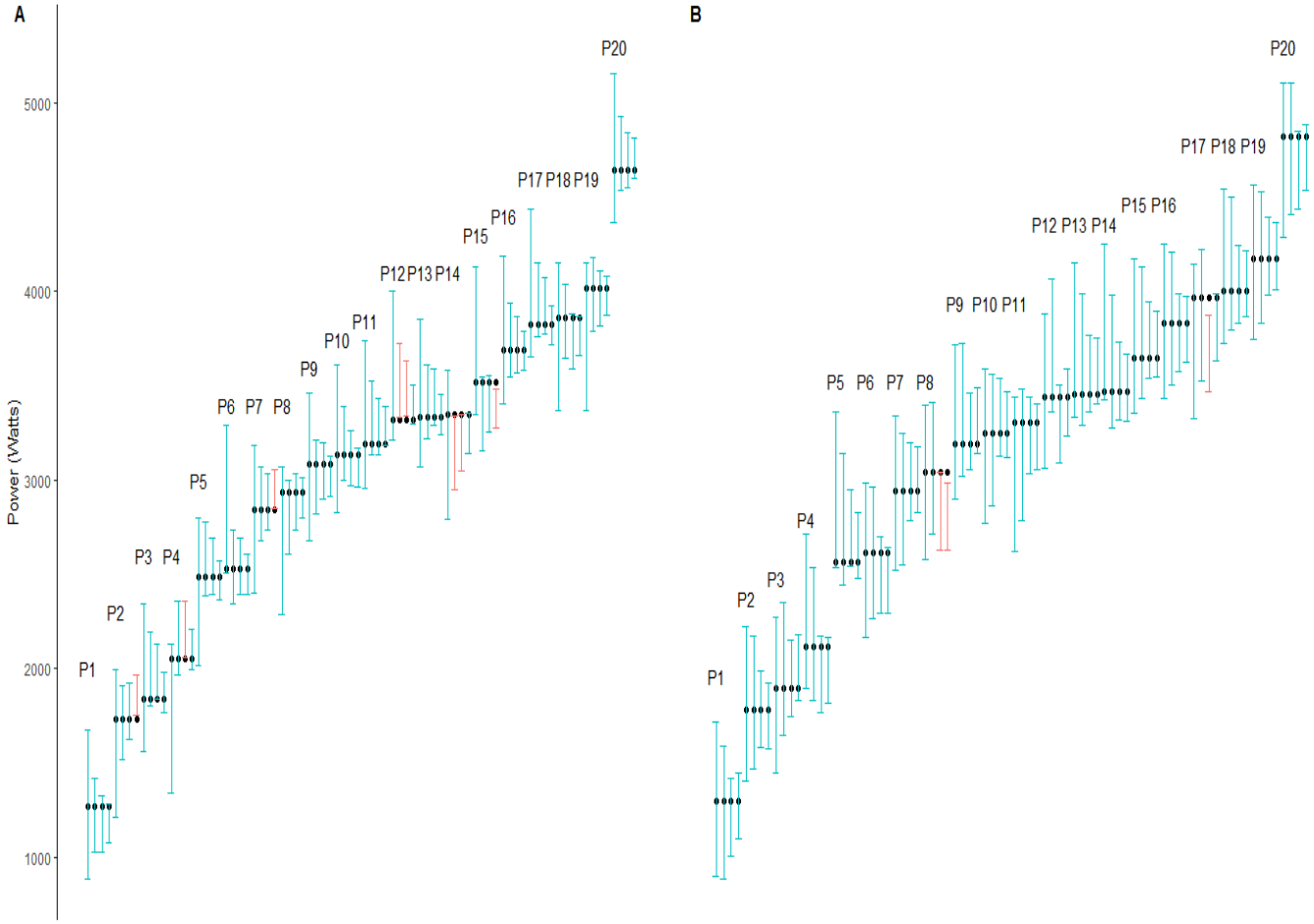
2.4 Framework extension 1 – Mean of multiple measurements to reduce confidence interval widths.

The first extension we introduce to reduce uncertainty is the simple practice of taking the mean of multiple test scores from the same individual. We start by ignoring potential differences in variation of observed scores obtained within and between testing sessions (e.g. intra- and inter-day reliability). With the proposed statistical model presented in section 2.1, the mean of multiple independent test scores $(\bar{O}_s)$ collected from a single individual $(O_{s_1}, O_{s_2}, \ldots, O_{s_n})$ have their own normal sampling distribution with mean equal to T_s and standard deviation equal to $\frac{\sigma}{\sqrt{n}}$, where n is the number of measurements made to calculate the mean (details provided in the supplementary files). To construct CI's with the mean observed score from an individual the calculation becomes $\bar{O} \pm M \cdot \frac{\widehat{TE}}{\sqrt{n}}$, where M is the same multiple determined previously for the desired CI width and expected accuracy of $\widehat{\text{TE}}$. If for example, a test is repeated four times and the mean calculated, then the CI width will be halved $\left(\frac{\widehat{TE}}{\sqrt{4}} = \frac{\widehat{TE}}{2}\right)$. In figure 5 we can see the substantive decrease in 95% true score CI's with the mock data as the number of measurements increases.

It is often easier to make repeated measurements within a testing session compared with repeating the entire testing procedures across different days. This distinction, however, is important given variation in observed scores will generally be greater across testing sessions compared to within a session. A simple model to capture this difference is the hierarchical model $O_{s_{ij}} = T_s + b_i + \epsilon_{ij}$, where i refers to the day of the measurement, j is the specific measurement on day i, b_i is the random effect offset for day i, and ϵ_{ij} is the random within-day measurement error that is independent of b_i. This hierarchical model assumes that on a given day, observed scores will systematically deviate from the true score by an amount (random offset b_i) that can be described by a normal distribution with mean 0, and standard deviation σ_b. If repeated measurements are made on this day, they will then be distributed around the mean $T_s + b_i$, which can be modelled by a normal distribution with standard deviation σ_ϵ. As stated above, we expect $\sigma_b > \sigma_\epsilon$. Using this model, the total measurement error is $\sigma = \sqrt{\sigma_b^2 + \sigma_\epsilon^2}$, and we can show that taking the mean of single measurements across n days is distinct from taking the mean of m measurements within a single day (or m measurements within each n days). For a single measurement across n days the typical error is equal to $\sqrt{\frac{\sigma_b^2+\sigma_\epsilon^2}{n}}$ (see supplementary files). A lower reduction in the typical error equal to $\sqrt{\sigma_b^2 + \frac{\sigma_\epsilon^2}{m}}$ is obtained for the mean of m measurements within a single day. Finally, the greatest reduction is obtained with taking the mean of m measurements in a single day and then repeating this process across n days and taking the overall mean (e.g. grand mean), which provides a typical error equal to $\sqrt{\frac{\sigma_b^2}{n} + \frac{\sigma_\epsilon^2}{nm}}$ (see supplementary files). In our mock data set we assume that σ_b= 160 Watts and σ_ϵ=120 Watts, such that $\sqrt{\sigma_b^2 + \sigma_\epsilon^2}$ = 200 Watts. For simplicity we assume that the practitioners knows the inter- and intra-day measurement errors and we consider three scenarios: 1) mean of four measurements on a single day; 2) mean of four single measurements across four days; and 3) grand mean of four different measurements made on four different days. In scenario 1 the 95% true score CI equals $(O_{s_{11}} + O_{s_{12}} + O_{s_{13}} + O_{s_{14}})/4 \pm 1.96 \times 171$ Watts; in scenario 2 the 95% true score CI equals $(O_{s_{11}} + O_{s_{21}} + O_{s_{31}} + O_{s_{41}})/4 \pm 1.96 \times 100$ Watts; in scenario 3 the 95% true score CI equals

$(O_{s_{11}} + \ldots + O_{s_{14}} + O_{s_{21}} + \ldots + O_{s_{24}} + O_{s_{31}} + \ldots + O_{s_{34}} + O_{s_{41}} + \ldots + O_{s_{44}})/16 \pm 1.96 \times 85.4$ Watts. In the above example it is easy to see that if σ_b is substantively greater than on σ_ϵ then there may be minimal improvement in the width of true score CI's when taking multiple measurements in a single day.

Figure 5: Visualisation of true score confidence intervals calculated using the mean of different numbers of observations assuming a standard model (A) and a hierarchical model differentiating between inter- and intra-day reliability (B).



Data from 20 randomly selected participants (P1 to P20) are presented and grouped. Circles represent true scores and intervals represent 95% true score confidence intervals. Plot A: Intervals are calculated from the mean of 1, 4, 7 and 14 observations assuming a standard model with *TE* of 200 Watts. Plot B: Intervals are calculated across four scenarios (1: single observation; 2: mean of four measurements on a single day; 3: mean of a single measurement on four different days; 4: grand mean of four different measurements made on four different days) assuming a hierarchical model with inter-day measurement error of 160 Watts and intra-day measurement error of 120 Watts. For each cluster of points (participants), the shrinking magnitude of intervals can be observed. Intervals are centred on the mean of the observed scores and those that do not include the true score are highlighted in red.

3. ESTABLISHING PLAUSIBLE CHANGE SCORES

A key outcome process in sport and exercise is to investigate intervention efficacy, via assessing pre- to post-intervention change. This requires accounting for uncertainty in observed scores due to measurement error across all testing occasions (e.g. pre and post). In sport and exercise, improvements in physical qualities resulting from targeted interventions are often small in magnitude and measurement errors relatively large (Mengersen *et al,* 2016; Maughan *et al,* 2018; Hall and Kahan 2019). Therefore, it is important that uncertainty in observed change scores are accounted for to avoid misinterpretations of the data. Similar to methods used for quantifying uncertainty around baseline values, CI's can also be used to express uncertainty in pre-post change scores. In the following sub-sections, we will briefly recap methods outlined in Swinton *et al.* (2018), and then detail how intermediate (within-intervention) testing can be used to obtain more precise estimates of change across an intervention. Additionally, we will outline how intermediate testing can also be combined with the mean of multiple measurements to further enhance precision of estimates.

3.1 True score change confidence intervals

Within the examples provided herein we assume that measurement error is consistent across individuals in a group, and across the intervention, such that observed score variation is the same for both pre- and post-intervention scores. Under these assumptions, observed change scores ($OS_{change} = OS_{post} - OS_{pre}$) follow a normal distribution with mean equal to the true score change and standard deviation equal to $\sqrt{2}\sigma$ (which is estimated by $\sqrt{2}\widehat{\mathrm{TE}}$). Readers are referred to supplementary files for further explanation of this derivation. True score change CI's are obtained by applying an estimate of typical error around the observed pre- to post-score difference, such that the CI is generated from $OS_{\mathrm{change}} \pm \left(M \times \sqrt{2}\widehat{\mathrm{TE}}\right)$. We can see from this equation that if we want to have the same coverage for true score change that we used for baseline true scores (e.g. 95% CI's), then the intervals will be wider by a factor of $\sqrt{2}$. Within sport and exercise, an improvement just beyond zero for most biological markers or measures of physical capacity is practically meaningless. Therefore, it is recommended that a threshold value beyond zero is identified to judge whether an intervention is effective (Swinton *et al*, 2018). Whilst there is debate on how best to generate such a threshold (Bernards *et al,* 2017; Harvey 2019), one common method is the use of a smallest worthwhile change (SWC) value. This value is either selected subjectively based on prior experience with a particular group and delivering similar interventions, or obtained via more objective methods such as effect size calculations (e.g. Cohen's D) (Hopkins, 2004). Readers are directed to several sources for further information regarding SWC values (Copay *et al*, 2007; Ferreira

et al, 2012). Once a threshold is selected, an intervention is considered successful if both tails of the change score CI lie beyond the threshold in the desired direction. We recommended previously that a CI calculated with $OS_{\text{change}} \pm \widehat{\text{TE}}$ was appropriate and provided an approximate 50% true score change CI (Swinton *et al*, 2018). In many instances in sport and exercise, however, change score CI's (including that recommended) will be so wide, that individuals will be required to make improvements substantially beyond the threshold to reliably state an intervention was successful. As an example, research has demonstrated that with healthy individuals a strength and power training intervention may be expected to improve an individual's concentric vertical jump power by approximately 100 W (Oliver *et al,* 2013; Taylor *et al,* 2016). Research has also demonstrated, however, that $\widehat{\text{TE}}$ of mean vertical jump power is expected to be in the range of 50-200 W (Cormack *et al,* 2008; Taylor *et al,* 2016). Therefore, if a threshold of 100 W was selected to judge the intervention as a success, then the observed change score using our previous recommendation would have to exceed 150 Watts (using $\widehat{\text{TE}}$ =50 Watts) or as much as 300 Watts (using $\widehat{\text{TE}}$ =200 Watts). Furthermore, if we selected a 95% true score change CI with $\widehat{\text{TE}}$ =200 Watts, then we would require an improvement of approximately $100 + 1.96\sqrt{2} \times 200 \sim 650$ Watts. This example highlights the need for methods that can reduce change score CI widths.

3.2 Framework extension 2 – Intermediate (within-intervention) testing

The second extension to the existing framework proposes collection of observed test scores at intermediate points (such as the mid-point) across an intervention. As will be demonstrated, by including intermediate testing points the observed change required to be confident that an intervention has resulted in true score change beyond the threshold selected can be reduced. Across many moderate to long-duration interventions in sport and exercise (i.e. 3-6 months), it may be reasonable to include for example two to ten intermediate testing occasions. In the following extension that we present, we make three key assumptions: 1) true score change across the intervention period is linear; 2) measurement error across the intervention remains consistent and is described by the same normal distribution with mean 0 and standard deviation σ; and 3) measurement errors at each testing occasion are independent. Later, however, we will discuss an approach that does not require the final assumption, which is most relevant when many intermediate measurements are made. Each of these modelling assumptions can be appropriate within a single intervention; however, the first assumption is perhaps the most challenging given response to many interventions may not be linear or follow a simple dose-response profile. To account for violations in each of these assumptions the analysis process becomes substantially more complex, and the purpose of this review is to present a set of relatively simple and intuitive

methods that can be used and clearly understood by practitioners and applied researchers. Like all statistical analyses, simplifications and modelling assumptions are used, and so the user must assess the extent to which the assumptions are appropriate within their own interventions and contexts.

3.3 Monte Carlo simulation

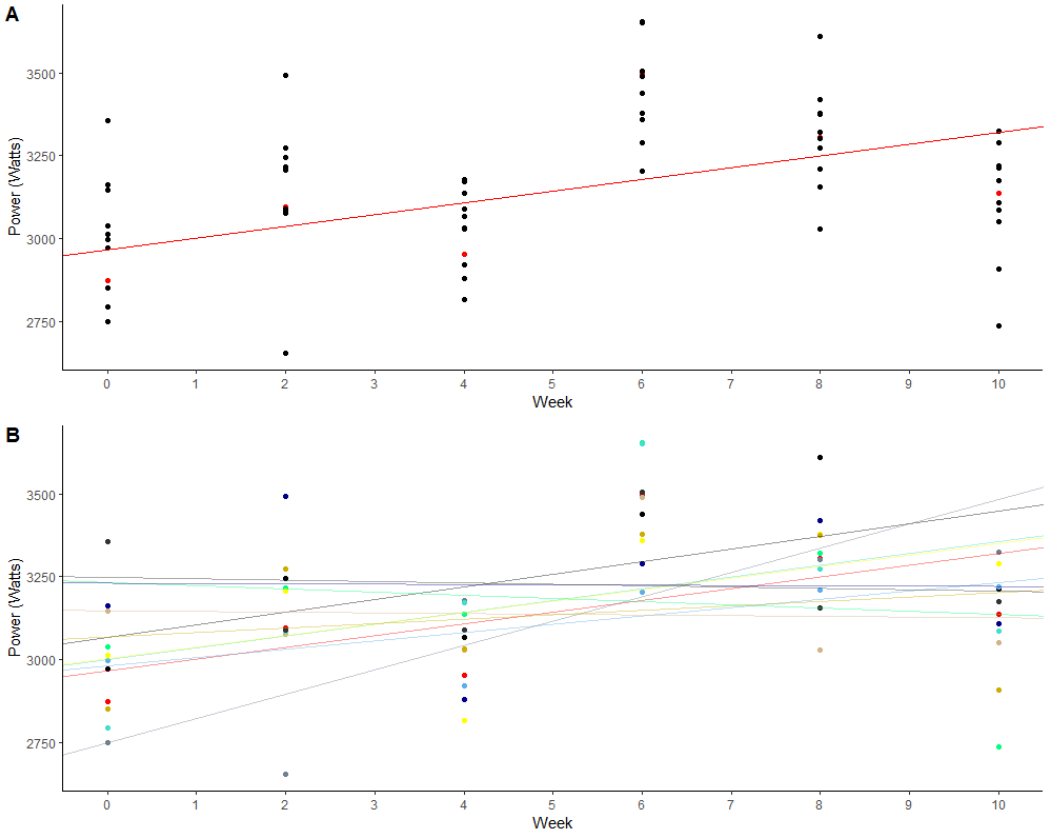
We start by providing a more conceptual overview of the extension. If the assumptions of linearity, and consistent and independent errors are accepted, we can conceptualise fitting a regression line through the observed measurements to estimate the rate of change and subsequently the absolute change over an interpolated interval. Based on the assumptions outlined, each observed measurement follows a distribution based on the rate of change and the typical error. With each additional data point there will be less influence of measurement error and subsequently more precise estimates of the true underlying rate of change can be made.

The simplest way to conceptualise the extension is to view the process as a Monte Carlo simulation. Monte Carlo approaches represent flexible experimental tools that can artificially create and thereby study and estimate sampling distributions (Myers, Ahn and Jin 2011). If we consider an individual completing a resistance training intervention to improve power with measurements taken each week, we can estimate any change with a regression line of the form: Power$= m \times week + c;$ where m is the weekly change in power (the gradient) and c is the power at week 0 (the intercept). Given our assumptions each measurement comprises the true score and measurement error from a normal distribution which can be modelled with mean zero and standard deviation represented with a known standard deviation for simplicity. The Monte Carlo approach is executed by simply adding random error values from our distribution to each observed point and then fitting a new regression line. By repeating this process again and again (each time adding new random draws from the normal distribution), a sample of regression lines will be obtained (Figure 6). The smaller the measurement error and the more data points that are included, the more consistent the regression lines will be. For each regression line we multiply the gradient by the number of weeks to obtain the estimated change score. We then obtain a CI for this change by arranging the estimates in order and selecting for example the middle 68% or the middle 95%.

In Figure 6 we illustrate the case where we have an individual with a true baseline value of 3000 Watts, a true change score of 30 Watts per week and known measurement error of 200 Watts. The red points in Figure 6 are the values we observe at each time point and the regression line they create over the ten weeks. Figure 6 also illustrates ten

simulations around each observed score and the new regression lines they create. If we perform the Monte Carlo simulation for 10,000 iterations we that find that close to a 75% CI will exceed 100 Watts which we select as the SWC (75% CI: 75 to 628 Watts). In the following section we provide an analytical method that provides the same results as the Monte Carlo simulation and can be used to help determine how many intermediate testing points may be required to match likely changes, the SWC, and assumed typical errors.

Figure 6: Visualisation of Monte Carlo simulation quantifying uncertainty in true score change using intermediate testing.



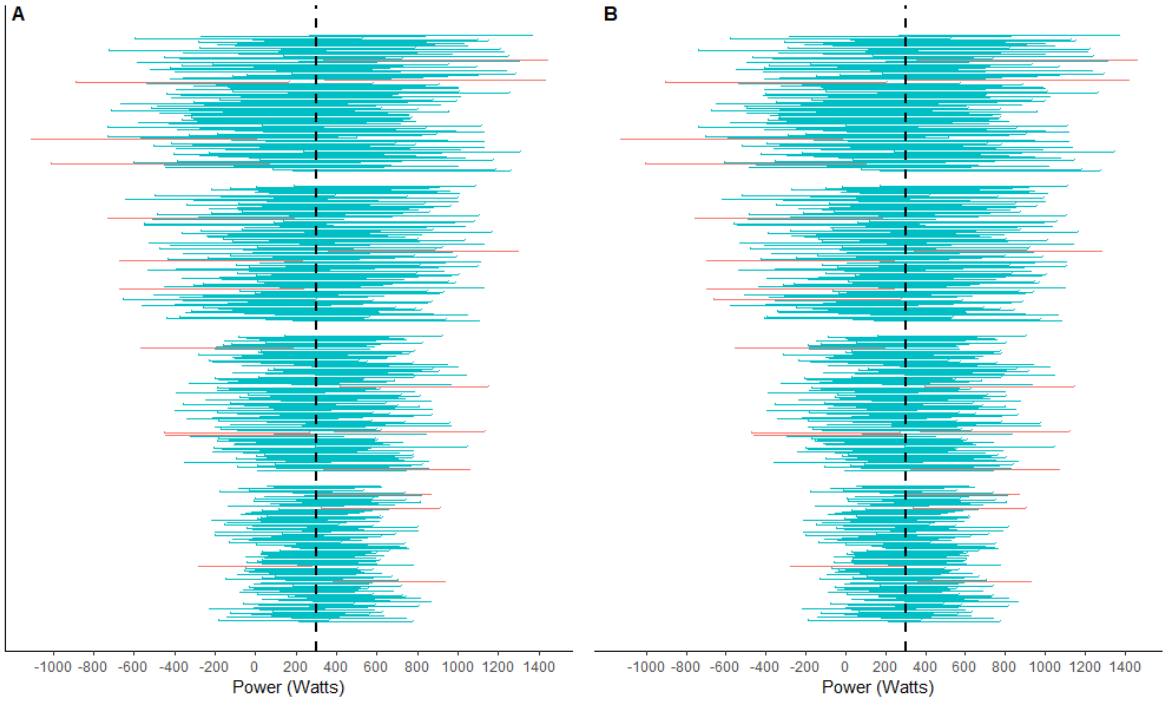
Data from a single participant. Red points represent observed data and all other points represent Monte Carlo simulation with normal distribution and TE of 200 Watts around observed. Red line represents linear regression estimate of true score change across intervention from observed data. Other lines represent regression lines from simulated data. Different slopes of regression lines illustrate uncertainty in estimate using observed data.

3.3 Least squares regression calculations

Whilst the previous section provides an intuitive approach to uncertainty in regression modelling, a more practical and potentially useful approach is obtained analytically using ordinary least squares estimators. First, we introduce notation used for linear regression. The standard linear regression model is generally expressed as $\boldsymbol{y} = \boldsymbol{X\beta} + \boldsymbol{\epsilon}$, where $\boldsymbol{y}$ is the vector of observed scores, $\boldsymbol{X}$ is referred to as the design matrix and includes information regarding the time points of each measurement, $\boldsymbol{\beta}$ is the regression coefficients which include the origin and slope, and $\boldsymbol{\epsilon}$ refers to the error variable describing the expected spread of values from the regression line. To obtain a straight line the design

matrix requires two columns, the first is a series of 1's to obtain the intercept, and the second is the time points of the measurements (e.g. 0, 2, 4, …10 for weeks). The ordinary least squares estimator of $\boldsymbol{\beta}$ is given by $\widehat{\boldsymbol{\beta}} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}$, and has sample variance $\sigma^2(\boldsymbol{X}^T\boldsymbol{X})^{-1}$ (see supplementary files for more details). Therefore, once we have our data and typical error estimate, we can estimate the regression slope $\widehat{\boldsymbol{\beta}}_{1,2}$ (2nd row of the column vector) and its standard error $\sqrt{\sigma^2(\boldsymbol{X}^T\boldsymbol{X})^{-1}_{2,2}}$ (square root of 2nd row 2nd column of 2x2 matrix). We then calculate a CI by replacing σ^2 with $\widehat{\mathrm{TE}}$ and selecting the same multiple as outlined in section 2.3 to obtain a given CI width. As a practical example, we use ordinary least squares regression on the data obtained in the previous Monte Carlo simulation and show their correspondence (Figure 7). The six observed score values were: $\boldsymbol{y} = [2874, 3097, 2953, 3499, 3305, 3136]^T$ and we have the design matrix $\boldsymbol{X} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 2 & 4 & 6 & 8 & 10 \end{bmatrix}^T$. Plugging in σ^2 = 200 Watts gives $\widehat{\boldsymbol{\beta}} = [2967 \quad 35.4]^T$ and $\sigma^2(\boldsymbol{X}^T\boldsymbol{X})^{-1} = \begin{bmatrix} 20952 & 2857.1 \\ -2857.1 & 571.4 \end{bmatrix}$. To obtain an approximate 75% CI we have $10 \times \left(35.4 \pm 1.15\sqrt{571.4}\right) = 79.4$ to 629 Watts, which we can see aligns with the Monte Carlo simulation.

Figure 7: Visualisation of 95% true score change confidence intervals calculated using ordinary least squares regression combined with Monte Carlo simulation (A) or direct standard error calculation (Right).



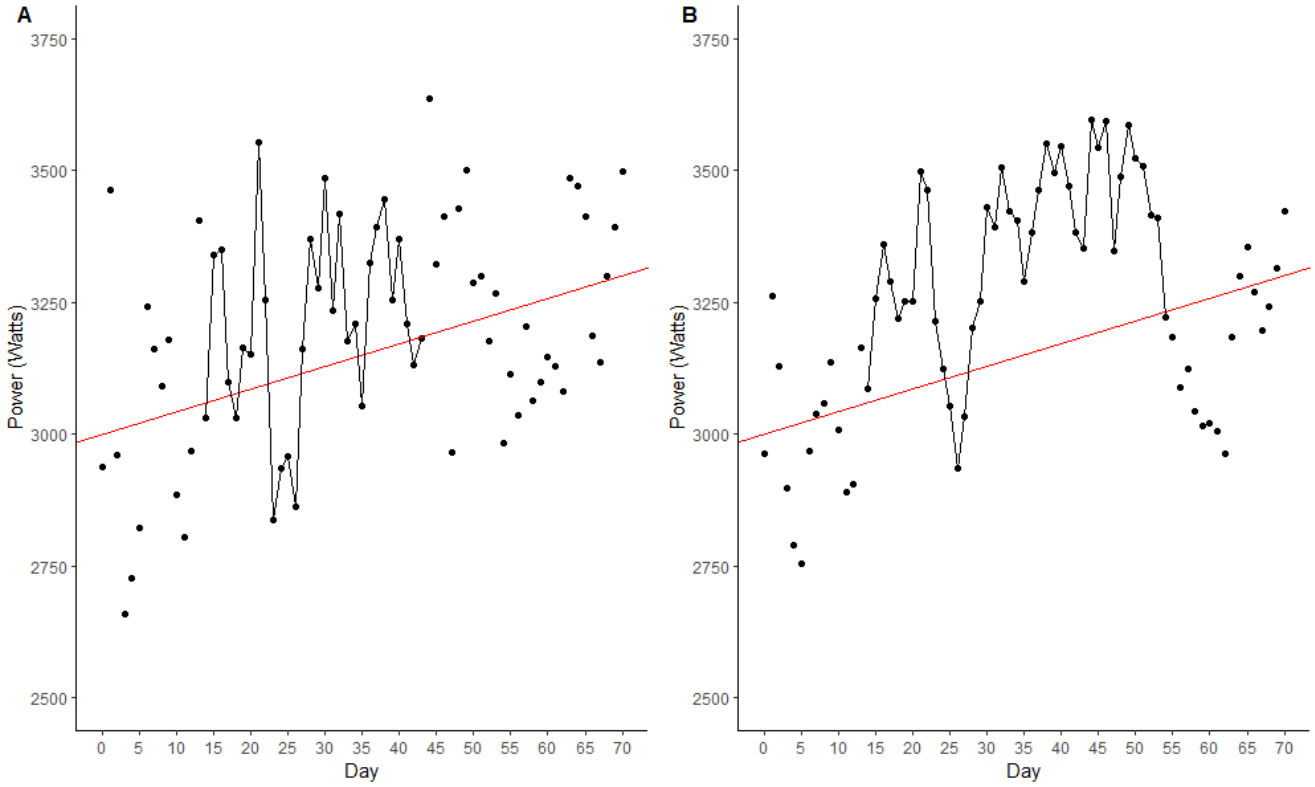
One hundred 95% true score change confidence are created from each set of regressions featuring increasing number of data points from top to bottom. Clusters are obtained from data generated with 3, 6, 11, and 21 data points and visualise shrinking magnitudes with increasing data. Calculations are made assuming the *TE* of 200 Watts is known. Vertical lines show the true score change across the intervention. Intervals that do not include the true score are highlighted in red. Extreme similarity in results highlight equivalence in approaches.

3.4 Autoregressive modelling

In some instances, we may be able to collect data at high frequencies such that multiple measurements are collected weekly or multiple times per week. For example, it has become relatively common to perform tests such as the vertical jump prior to each training session (Greig *et al*, 2020). Tests such as the vertical jump are non-fatiguing and can be used to collect a range of variables to monitor readiness and fitness attribute including power, rate of force development and force production at high velocities (Greig *et al*, 2020). Similarly, body composition measurements including the use of bioelectrical impedance to assess fat and fat-free mass can be completed multiple times each day. In these instances, the assumption that measurement errors are independent is less tenable, and capturing associations may lead to more robust CI estimates. A relatively simple model that can be used is linear regression $\boldsymbol{y} = \boldsymbol{X\beta} + \boldsymbol{\xi}$, where errors $\boldsymbol{\xi}$ follow an autoregressive process such that the magnitude and direction are associated with those obtained previously. The simplest autoregressive process is the AR(1) model with serial correlation parameter θ ($|\theta| < 1$), which sets correlations among errors equal to θ^k, where k is the difference in time between the two errors (see supplementary files for further details). In more detail, the AR(1) process for errors can be expressed as $\xi_i = \theta\xi_{i-1} + \psi_i$, where ψ_i is normally distributed with mean 0 and variance ζ^2. Additionally, it can be shown that the overall variance of the AR(1) process $Var(\xi_i)$ is equal to $\frac{\zeta^2}{1-\theta^2}$. And so, if we set $\frac{\zeta^2}{1-\theta^2} = \sigma^2$ then under the AR(1) model with n measurements, the observed scores $\boldsymbol{y}$ are distributed as $N(\boldsymbol{X\beta}, \boldsymbol{R}\sigma^2)$, where $\boldsymbol{R}$ is a $n \times n$ correlation matrix with entry $i, j = \theta^{|j-i|}$. There are two potential options when using linear regression with an AR(1) model, the user can: 1) use specialist software to fit the model directly and estimate all parameters including θ and ζ; or 2) input the estimated typical error and assume a value for θ that is likely to be reasonable. In Figure 8 we illustrate daily power measurements from a single participant simulated from an AR(1) model with weekly increase of 30 Watts, σ=200 Watts, and θ=0.2 and θ=0.8 (e.g. ζ^2 is set accordingly). As θ increases, Figure 8 illustrates greater association between random errors and subsequent slower 'walk' that occurs as observed scores oscillate around true scores. Using the arima function in R, the estimated improvement and 95% CI for the two models equal 284 [95% CI: 77 to 490 Watts] and 320 [95% CI: -80 to 720 Watts] for θ=0.2 and θ=0.8, respectively. To obtain direct calculations we use ordinary least squares estimators as they provide formula-based statistics and avoid generalized least squares estimation that gives the best and unbiased estimators but requires an iterative process (Tang and Landes 2020). The ordinary least squares estimator for the slope $\hat{\beta}_1$ remains unchanged from the standard model ($\widehat{\boldsymbol{\beta}} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}$), whereas the sample variance changes to $\sigma^2(\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{R}\boldsymbol{X}(\boldsymbol{X}^T\boldsymbol{X})^{-1}$ (see supplementary files for further details). Plugging in the

known measurement error variance and calculating $\boldsymbol{R}$ from the known serial correlation values, we obtain similar estimated improvements and 95% CI's of 273 [95% CI: 80 to 466 Watts] and 263 [95% CI: -166 to 694 Watts] for θ=0.2 and θ=0.8, respectively. From the example above, we can see that true score change CI's will generally be lower for lower values of θ when using the AR(1) model.

Figure 8: Visualisation of linear regression modelling increase in power over intervention with autoregressive errors and correlations equal to 0.2 (A) and 0.8 (B)



Red regression lines show true scores and the linear improvement across the intervention. Observed scores are illustrated by data points. Section in each plot highlighted with dashed line illustrates the autoregression of errors with a slower 'walk' either side of the regression line when θ=0.8 (B) compared with θ=0.2 (A).

3.3 Combining framework extensions 1 and 2

In the previous sections we have shown that two different methods including taking the average of multiple measurements and completing intermediate testing with regression modelling via Monte Carlo simulations, least squares estimators or autoregressive modelling, can substantively reduce uncertainty in measurements, thus allowing for better informed decisions regarding intervention selection, monitoring and evaluation. We finish this review, by highlighting that further reductions in uncertainty can be made if the two methods are combined. In section 3.3 we showed that intermediate testing can still result in relatively wide CI's if measurement error is large and especially if we were to generate large true score change % CI's such as 95% CI's. In the example in section 3.3, we would obtain

a 95% true score change CI of $10 \times \left(35.4 \pm 1.96\sqrt{571.4}\right) = -115 \text{ to } 823$ Watts if we were to fit a regression with one measurement every two weeks (Note this is an improvement on the 95% CI = -200 to 908 Watts that would be obtained with only a pre and post measurement). If in addition to the intermediate testing, a practitioner for example duplicated the testing sessions across two daily sessions and took the average of four measurements within each session (assuming σ_b= 160 and σ_ϵ=120 Watts), then σ reduces from 200 Watts to 121 Watts and so our 95% true score change CI reduces to $10 \times \left(35.4 \pm 1.96\sqrt{208.6}\right) = 71 \text{ to } 637$ Watts (see supplementary files for full details).

As a final note, the examples above highlight the importance in differentiating between selecting the peak value within a test, and the maximum value obtained across a series of tests. The former is often appropriate where the temporal nature of a test is pertinent and peak values describe a unique phenomenon that is not captured well with the mean. Here a trade-off may occur with improved information on the client, but increased measurement error (e.g. measurement error is likely to be larger when measuring peak power during a vertical jump compared to mean power). In contrast, the latter (selecting maximum value across a series of tests) is not appropriate if we assume that test performance is suitably modelled with a normal distribution centred on the true score. Despite the goal of many tests in sport and exercise to quantify maximum performance, if the maximum value of a series of tests is selected as the measure for evaluation, this will almost certainly be an overestimate generated from a positive random error. In contrast, the mean of multiple test measurements is unbiased and has the advantageous property that measurement error is reduced to $\frac{\sigma}{\sqrt{n}}$. For these reasons, practitioners should not for example select the 'peak of peaks' when evaluating a client.

4 SUMMARY

Measurement to inform the selection, monitoring and evaluation of interventions has become integral in sport and exercise. Frequently, however, uncertainty in measurements are not captured such that the above processes are likely to be limited leading to erroneous conclusions and poor decision making in too many cases. Even when attempts are made to rigorously quantify uncertainty in practice, basic approaches such as the use of typical error combined with a single measurement or the use of pre- and post-intervention data only, may lead to limited information and subpar decision making. This is due to many measurements commonly used in sport and exercise encompassing large errors due to instrumentation and biological noise. In these situations, the answer should not be to abandon attempts at quantifying uncertainty, but instead to adopt measurement practices that lower measurement error to levels that facilitate good decision making. In this review, we have discussed two such general approaches that are likely to be effective in many circumstances, namely repeated testing and regression modelling. We have also attempted in this review to balance statistical rigour and the need to adopt practices that are pragmatic, easy to use, and relatively easy to understand. More complex and in some circumstances more rigorous statistical procedures could be used and "hidden" behind software that the user does not have to fully appreciate. Despite the trade-offs, we believe that the simpler approaches presented here provide a strong foundation and will be of use to many practitioners. Based on the text provided and additional detail in supplementary files, practitioners and applied researchers can implement these procedures using simple spreadsheets and adapt the R code provided for greater flexibility.

REFERENCES

1. Bernards JR, Sato K, Haff GG, Bazyler CD. Current research and statistical practices in sport science and a need for change. Sports. 2017 Nov 15;5(4):87. https://doi.org/10.3390/sports5040087.
2. Copay AG, Subach BR, Glassman SD, Polly Jr DW, Schuler TC. Understanding the minimum clinically important difference: a review of concepts and methods. The Spine Journal. 2007 Sep 1;7(5):541-6. https://doi.org/10.1016/j.spinee.2007.01.008.
3. Cormack SJ, Newton RU, McGuigan MR, Doyle TL. Reliability of measures obtained during single and repeated countermovement jumps. International journal of sports physiology and performance. 2008 Jun 1;3(2):131-44. https://doi.org/10.1123/ijspp.3.2.131.
4. Ferreira, M.L., Herbert, R.D., Ferreira, P.H., Latimer, J., Ostelo, R.W., Nascimento, D.P. and Smeets, R.J., 2012. A critical review of methods used to determine the smallest worthwhile effect of interventions for low back pain. *Journal of clinical epidemiology*, *65*(3), pp.253-261. https://doi.org/10.1016/j.jclinepi.2011.06.018.
5. Greig L, Stephens Hemingway BH, Aspe RR, Cooper K, Comfort P, Swinton PA. Autoregulation in resistance training: addressing the inconsistencies. Sports medicine. 2020 Nov;50(11):1873-87. https://doi.org/10.1007/s40279-020-01330-8.
6. Hall KD, Kahan S. Maintenance of lost weight and long-term management of obesity. Medical Clinics. 2018 Jan 1;102(1):183-97. https://doi.org/10.1016/j.mcna.2017.08.012.
7. Harvey LA. A minimally important treatment effect is a key but elusive concept. Spinal Cord. 2019 Feb;57(2):83-4. https://doi.org/10.1038/s41393-019-0241-0.
8. Hopkins WG. Measures of reliability in sports medicine and science. Sports medicine. 2000 Jul;30(1):1-5. https://doi.org/10.2165/00007256-200030010-00001.
9. Hopkins WG. How to interpret changes in an athletic performance test. Sportscience. 2004 Jan 1;1-7. https://www.sportsci.org/jour/04/wghtests.htm
10. Maughan RJ, Burke LM, Dvorak J, Larson-Meyer DE, Peeling P, Phillips SM, Rawson ES, Walsh NP, Garthe I, Geyer H, Meeusen R. IOC consensus statement: dietary supplements and the high-performance athlete. International journal of sport nutrition and exercise metabolism. 2018 Mar 1;28(2):104-25. doi: https://doi.org/10.1136/bjsports-2018-099027.
11. Mengersen KL, Drovandi CC, Robert CP, Pyne DB, Gore CJ. Bayesian estimation of small effects in exercise and sports science. PloS one. 2016 Apr 13;11(4):e0147311. https://doi.org/10.1371/journal.pone.0147311.

12. Myers ND, Ahn S, Jin Y. Sample size and power estimates for a confirmatory factor analytic model in exercise and sport: A Monte Carlo approach. Research quarterly for exercise and sport. 2011 Sep 1;82(3):412-23. https://doi.org/10.1080/02701367.2011.10599773.
13. Oliver JM, Jagim AR, Sanchez AC, Mardock MA, Kelly KA, Meredith HJ, Smith GL, Greenwood M, Parker JL, Riechman SE, Fluckey JD. Greater gains in strength and power with intraset rest intervals in hypertrophic training. The Journal of Strength & Conditioning Research. 2013 Nov 1;27(11):3116-31. https://doi.org/10.1519/JSC.0b013e3182891672.
14. Swinton PA, Hemingway BS, Saunders B, Gualano B, Dolan E. A statistical framework to interpret individual response to intervention: paving the way for personalized nutrition and exercise prescription. Frontiers in Nutrition. 2018 May 28;5:41. https://doi.org/10.3389/fnut.2018.00041.
15. Tang J, Landes RD. Some t-tests for N-of-1 trials with serial correlation. Plos one. 2020 Feb 4;15(2):e0228077. https://doi.org/10.1371/journal.pone.0228077.
16. Taylor KL, Hopkins WG, Chapman DW, Cronin JB. The influence of training phase on error of measurement in jump performance. International Journal of Sports Physiology and Performance. 2016 Mar 1;11(2):235-9. https://doi.org/10.1123/ijspp.2015-0115.

Doi:| SportRχiv Preprint version 1

Statistical Methods to Reduce the Effects of Measurement Error in Sport and Exercise: A Guide for Practitioners and Applied Researchers

Paul A. Swinton[1], Ben Stephens Hemingway[1], Iain J Gallagher[2], Eimear Dolan[3]

Supplementary file 1: Additional mathematical detail

Section 2.2. Estimating standard deviation of measurement error.

Given $O_s \sim N(T_S, \sigma^2)$, where O_s is an observed score, T_S is the true score, and σ is the standard deviation of the measurement error, then the sample standard deviation $\sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(O_{si} - \bar{O}_s)^2}$ is close to an unbiased estimator of σ^2 and can be used for $\widehat{TE}$ when we are able to collect many repeated measurements on an individual across a time period where the true score is not expected to change.

More likely, $\widehat{TE}$ is obtained by a test retest procedure on a group of individuals. Here we assume that individuals possess different true scores, but experience the same measurement error standard deviation. If we take the difference between two observed scores $O_{s_2} - O_{s_1}$ and we assume that the errors are independent, then $O_{s2} - O_{s_1} \sim N(0, 2\sigma^2)$ from $Var(X_2 - X_1) = Var(X_2) + Var(X_1) = 2\sigma^2$.

Calculating the standard deviation of the observed difference scores provides an estimate of $\sqrt{2}\sigma$. Hence to obtain an estimate of σ, we divid the standard deviation of the observed difference scores by $\sqrt{2}$.

Section 2.3 Constructing true score confidence intervals

For the calculation of CI's it is useful to introduce additional notation and concepts. The first is the notation: $100(1-\alpha)\%$, which describes the width of the CI. Here, α is a variable that we choose to set the interval and importantly link the width of the CI to the correct multiple of σ if known, or our estimate of σ which is expressed as $\widehat{TE}$. To set a 90% CI for example, then α must be set to $\alpha = 0.1$ to give $100(1-0.1)\% = 90\%$. Given the typical assumption that observed scores are normally distributed we evoke the relevant properties of the distribution, such that a $100(1-\alpha)\%$ CI for true score is obtained with $O_s \pm \sigma \times Z_{(1-\alpha/2)}$. The coefficient $Z_{(1-\alpha/2)}$ is referred to as the $(1-\alpha/2)$-th quantile of the standard normal distribution. In our example where we set α to 0.1 (i.e. for a 90% confidence interval), we require $Z_{(1-0.1/2)}$, or the 0.95th quantile of the standard normal distribution. To obtain this value we can look up standard statistical tables or use the following code in R qnorm$(1-\alpha/2)$. Using these

methods, we find that $Z_{0.95}$ is equal to 1.64 (qnorm(0.95)) and so a 90% true sore CI for an individual would equal $O_s \pm \sigma \times 1.64$.

It is important to acknowledge that in practice we never know σ and studies only report imperfect estimates $\widehat{\text{TE}}$, where accuracy will depend primarily on the number of individuals (or number of repeated trials) used in a test-retest. To account for this additional uncertainty, we use the $(1-\alpha/2)$-th quantile value from a t-distribution which is similar in shape to the normal distribution but has heavier tails (i.e. greater proportion of values away from the centre). The specific t-distribution is based on numbers used in our σ estimate and we say that it has degrees of freedom equal to $n-1$. For example, if we estimate σ using 20 participants ($n=20$), we would obtain a 90% true score CI with $O_s \pm \widehat{TE} \times t_{19,0.95}$, (i.e. the 0.95th quantile of the t-distribution with 19 degrees of freedom). Looking up statistical tables or using the R code (qt($1-\alpha/2$,df) or here qt(0.95,19), we find that $t_{19,0.95} = 1.73$ and so our 90% true score CI is calculated with $O_s \pm \widehat{TE} \times 1.73$. Alternatively for example, if we wanted to calculate a 50% true score CI with the t-distribution, we would set $\alpha = 0.5$, $t_{19,0.75} = 0.69$ to give $O_s \pm \widehat{TE} \times 0.69$. What is important to note, is that as the number of individuals increases the t-distribution approaches the normal distribution such that the coefficients used to multiply the $\widehat{TE}$ converge.

Section 2.4 Framework extension 1 – Mean of multiple measurements to reduce confidence interval widths

Given $O_s \sim N(T_S, \sigma^2)$ and the assumption that observed scores are independent, then taking the mean of n observed scores will give expectation

$$E(\bar{O}_s) = E\left(\frac{1}{n}\left(O_{s_1} + O_{s_2} + \cdots + O_{s_n}\right)\right) = \frac{1}{n}\left(E\left(O_{s_1}\right) + E\left(O_{s_2}\right) + \cdots + E\left(O_{s_n}\right)\right) = \frac{1}{n}(nT_S) = T_S.$$

We will also have variance

$$Var(\bar{O}_s) = Var\left(\frac{1}{n}\left(O_{s_1} + O_{s_2} + \cdots + O_{s_n}\right)\right) = \frac{1}{n^2}\left(Var\left(O_{s_1}\right) + Var\left(O_{s_2}\right) + \cdots + Var\left(O_{s_n}\right)\right) = \frac{1}{n^2}(n\sigma^2) = \frac{\sigma^2}{n}.$$

As a result we have $\bar{O}_s \sim N\left(T_S, \frac{\sigma^2}{n}\right)$, such that the standard deviation will equal $\frac{\sigma}{\sqrt{n}}$ and CI's are constructed with $\bar{O} \pm \frac{\sigma}{\sqrt{n}} \times Z_{(1-\alpha/2)}$ or in practice $\bar{O} \pm \frac{\widehat{TE}}{\sqrt{n}} \times t_{(1-\alpha/2,df)}$.

Section 2.4 Framework extension 1 – Mean of multiple measurements in hierarchical model

In section 2.4 we set out the hierarchical model $O_{s_{ij}} = T_s + b_i + \epsilon_{ij}$, where i refers to the day of the measurement, j is the specific measurement on day i, b_i is the random effect offset for day i, and ϵ_{ij} is the random within-day measurement error that is independent of b_i. This hierarchical model assumes that on a given day, observed scores will systematically deviate from the true score by an amount that can be described by a normal distribution with mean 0, and standard deviation σ_b. If repeated measurements are made on this day, they will then be distributed around the mean $T_s + b_i$, which can be modelled by a normal distribution with standard deviation σ_ϵ. We expect $\sigma_b > \sigma_\epsilon$.

We can see that the expectation is

$$E\left(O_{s_{ij}}\right) = E\left(T_s + b_i + \epsilon_{ij}\right) = T_s.$$

We also have variance

$$Var\left(O_{s_{ij}}\right) = Var\left(T_s + b_i + \epsilon_{ij}\right) = Var(b_i) + Var\left(\epsilon_{ij}\right) = \sigma_b^2 + \sigma_\epsilon^2.$$

If we take two measurements on separate days then we have $Cov\left(O_{ij}, O_{i'j}\right) = 0$, and if we take two measurements on the same day we have $Cov\left(O_{ij}, O_{ij'}\right) = \sigma_b^2$.

And so, if we collect a single data point on n separate days and take the mean we have

$$Var\left(\frac{1}{n}\sum_{i=1}^{n} O_{ij}\right) = \frac{Var\left(O_{ij}\right)}{n} = \frac{\sigma_b^2 + \sigma_\epsilon^2}{n}.$$

If we take m data points on a single day and take the mean we have

$$Var\left(\frac{1}{m}\sum_{j=1}^{m} O_{ij}\right) = \frac{1}{m^2}\left[\sum_{j=1}^{m} Var(O_{ij}) + \sum_{j=1}^{m-1}\sum_{k=j+1}^{m} Cov\left(O_{ij}, O_{ik}\right)\right] =$$

$$\frac{1}{m^2}\left[mVar(O_{ij}) + (m^2 - m)Cov\left(O_{ij}, O_{ik}\right)\right] = \frac{\sigma_b^2 + \sigma_\epsilon^2}{m} + \frac{(m-1)}{m}\sigma_b^2 = \sigma_b^2 + \frac{\sigma_\epsilon^2}{m}.$$

If we take the mean of m measurements over n days, and then take the mean of this (e.g. the grand mean), the variance can be expressed as

$$Var\left(\frac{1}{n}\left(\sum_{i=1}^{n}\left(\frac{1}{m}\sum_{j=1}^{m} O_{ij}\right)\right)\right).$$

Note that $\bar{O}_{ij}$ is independent from $\bar{O}_{i'j}$ (e.g. mean of measurements on one day is independent from another), and so

$$Var\left(\frac{1}{n}\left(\sum_{i=1}^{n}\left(\frac{1}{m}\sum_{j=1}^{m} O_{ij}\right)\right)\right) = \frac{1}{n^2}\left(n\,Var\left(\frac{1}{m}\sum_{j=1}^{m} O_{ij}\right)\right) = \frac{\sigma_b^2}{n} + \frac{\sigma_\epsilon^2}{nm}.$$

Section 3.1 True score change confidence intervals

If we observed pre- (OS_{pre}) and post-intervention (OS_{post}) scores distributed as $OS_{pre} \sim N(TS_{pre}, \sigma^2)$ and $OS_{post} \sim N(TS_{post}, \sigma^2)$ and we assume that errors are independent, then $OS_{change} = OS_{post} - OS_{pre}$. The variance can be expressed as $Var\left(OS_{change}\right) = Var\left(OS_{post}\right) + Var\left(OS_{pre}\right) = 2\sigma^2$. Hence the standard deviation is $\sqrt{2}\sigma$ which we estimate as $\sqrt{2}\widehat{\mathrm{TE}}$. From our derivation in section 2.3, we would then create confidence intervals with $OS_{change} \pm \sqrt{2}\sigma \times Z_{(1-\alpha/2)}$ or in practice $OS_{change} \pm \sqrt{2}\widehat{\mathrm{TE}} \times t_{(1-\alpha/2,df)}$.

Note, if our pre- and post-intervention scores were obtained as the mean of n repeated measurements, then we would have $\overline{OS}_{pre} \sim N(TS_{pre}, \frac{\sigma^2}{n})$ and $\overline{OS}_{post} \sim N(TS_{post}, \frac{\sigma^2}{n})$ such that our confidence intervals would be created with

$\left(\overline{OS}_{post} - \overline{OS}_{pre}\right) \pm \frac{\sqrt{2}\sigma}{\sqrt{n}} \times Z_{(1-\alpha/2)}$ or in practice $\left(\overline{OS}_{post} - \overline{OS}_{pre}\right) \pm \frac{\sqrt{2}\sigma}{\sqrt{n}} \times t_{(1-\alpha/2,df)}$.

Section 3.3 Least squares regression calculations

If we wish to fit a simple linear model to data with a dependent variable y, and a single independent variable x we express the relationship as $y_i = \beta_0 + \beta_1 x_i + \epsilon_i$ such that β_0 is the intercept (the predicted value of y when $x = 0$), β_1 is the slope (the change in y for a unit increase in x), and ϵ_i is the random error component allowing for variation in y for identical values of x. In order to fit this simple model we need at least two data points (x_1, y_1) and (x_2, y_2). Usually, however, we have more than two points and we describe the data using column vectors denoted in bold $\boldsymbol{x} = (x_1, \ldots, x_n)^T$, $\boldsymbol{y} = (y_1, \ldots, y_n)^T$. If we define $\mathbf{1}_n$ to be a column vector of n ones, we can write the simple regression model as $\boldsymbol{y} = \beta_0 \mathbf{1}_n + \beta_1 \boldsymbol{x} + \boldsymbol{\epsilon}$. More compactly, if we define the parameter vector $\boldsymbol{\beta} = (\beta_0, \beta_1)^T$ and the $n \times 2$ matrix $\boldsymbol{X}$ defined to have $\mathbf{1}_n$ as the first column and $\boldsymbol{x}$ as the second, then we can write the model as $\boldsymbol{y} = \boldsymbol{X\beta} + \boldsymbol{\epsilon}$. Note, for a single measurement at for example weeks $0,2,\ldots,10$ we would have the matrix $\boldsymbol{X} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 2 & 4 & 6 & 8 & 10 \end{pmatrix}^T$.

The distribution of $\boldsymbol{\epsilon}$ is multivariate normal such that each observation in $\boldsymbol{y}$ is independent and variance constant and equal to σ^2, to give $\boldsymbol{y} \sim N_n(\boldsymbol{X\beta}, \boldsymbol{I}\sigma^2)$. The process of least squares allows us to estimate $\boldsymbol{\beta}$ with $\widehat{\boldsymbol{\beta}}$. This process requires us to define residuals $e_i = y_i - \hat{\beta}_0 + \hat{\beta}_1 x_i$. Residuals e_i are distinct from errors ϵ_i, however, they allow us to estimate the variability of the error terms and they are used to obtain $\widehat{\boldsymbol{\beta}}$. In this latter case, we obtain $\widehat{\boldsymbol{\beta}}$ such that it minimises the sum of the residuals squared. This sum of squares is expressed as $S(\widehat{\boldsymbol{\beta}}) = \sum_{i=1}^{n} e_i^2 = \boldsymbol{e}^T\boldsymbol{e} = (\boldsymbol{y} - \boldsymbol{X\widehat{\beta}})^T(\boldsymbol{y} - \boldsymbol{X\widehat{\beta}})$.

We find the least squares estimator by first differentiating $S(\widehat{\boldsymbol{\beta}})$ with respect to $\widehat{\boldsymbol{\beta}}$. This gives $\frac{\partial S(\widehat{\boldsymbol{\beta}})}{\widehat{\boldsymbol{\beta}}} = -2X^T\boldsymbol{y} + 2X^T\boldsymbol{X\widehat{\beta}}$. It follows that $\frac{\partial S(\widehat{\boldsymbol{\beta}})}{\widehat{\boldsymbol{\beta}}} = 0$, when $\widehat{\boldsymbol{\beta}} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}$.

The variance properties of $\widehat{\boldsymbol{\beta}}$ are contained in its covariance matrix, which in the case of simple linear regression is a 2×2 matrix. Here we have $Var(\widehat{\boldsymbol{\beta}}) = Var((\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}) = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T Var(\boldsymbol{y})((\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T)^T = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\sigma^2\boldsymbol{I_n}\boldsymbol{X}(\boldsymbol{X}^T\boldsymbol{X})^{-1} = \sigma^2(\boldsymbol{X^TX})^{-1}$.

We can obtain an unbiased estimator of σ^2 with $\hat{\sigma}^2 = \frac{1}{n-2}\sum_{i=1}^{n} e_i^2$.

When using the least squares regression approach to quantify improvement across an intervention under the assumptions that measurement error standard deviation remains constant and uncorrelated, and change is linear, we estimate improvement per unit time (e.g. per week) with $\hat{\beta}_1$ which is the second row entry of $\widehat{\boldsymbol{\beta}}$. If we have included time point 0 in $\boldsymbol{X}$, then change across the intervention is calculated with multiplying $\hat{\beta}_1$ by the final time point (e.g. $10 \times \hat{\beta}_1$ if week 10 was the final time point). We obtain a CI on the weekly improvement using the $2,2$ entry of $\sigma^2(\boldsymbol{X^TX})^{-1}$ and using the appropriate multiple. Where we assume σ is known, we have $\hat{\beta}_1 \pm \sigma^2(\boldsymbol{X^TX})^{-1}_{2,2} \times Z_{(1-\alpha/2)}$. We can then multiply our estimate and the lower and upper bounds by the final time point to obtain a true score change CI. Typically where we don't assume σ is known, we use our estimate $\widehat{\mathrm{TE}}$ and $\hat{\beta}_1 \pm \widehat{\mathrm{TE}}(\boldsymbol{X^TX})^{-1}_{2,2} \times t_{(1-\alpha/2,df)}$. With enough data, we may prefer to estimate σ directly and therefore use $\hat{\beta}_1 \pm \hat{\sigma}^2(\boldsymbol{X^TX})^{-1}_{2,2} \times t_{(1-\alpha/2,n-2)}$, where n is the number of data points used for the regression.

Section 3.3 Autoregressive model.

The final model discussed in this review is simple linear regression where the measurement errors follow an AR(1) process. In sport and exercise, if we are using this model we are potentially considering taking measurements multiple times per week, and so we can index according to days rather than weeks. The model can be expressed as $y_i = \beta_0 + \beta_1 x_i + \xi_i$, with errors ξ_i autocorrelated such that $\xi_i = \theta\xi_{i-1} + \psi$, where $|\theta| < 1$ is the serial correlation parameter, ψ is normally distributed with mean 0 and variance ζ^2. To derive the properties of the AR(1) errors it is best to express the series in-terms of an infinite-order moving average process $MA(\infty)$ where $\xi_i = \theta\xi_{i-1} + \psi_i = \theta(\theta\xi_{i-2} + \psi_{i-1}) + \psi_i = \theta(\theta[\theta\xi_{i-3} + \psi_{i-2}] + \psi_{i-1}) + \psi_i = \cdots = \theta^i\xi_0 + \sum_{j=0}^{i-1}\theta^j\psi_{i-j} \rightarrow \sum_{j=0}^{\infty}\theta^i\psi_{i-j}$ as $i \rightarrow \infty$ if $|\theta| < 1$ and ξ_0 is finite. By expressing the AR(1) series in this way we have $Var(\xi_i) = \sum_{j=0}^{\infty}\theta^{2i}\ \zeta^2$ which from an infinite geometric series $\sum_{j=0}^{\infty} ar^k = \frac{a}{1-r}$ for $|r| < 1$ and for series containing only even powers of r, $\sum_{j=0}^{\infty} ar^{2k} = \frac{a}{1-r^2}$. Thus $Var(\xi_i) = \sum_{j=0}^{\infty}\theta^{2i}\zeta^2 = \frac{\zeta^2}{1-\theta^2}$, $|\theta| < 1$.

If we set $\sigma^2 = \frac{\zeta^2}{1-\theta^2}$, then the linear regression model with AR(1) errors $\boldsymbol{y} = \boldsymbol{X\beta} + \boldsymbol{\xi}$ is distributed as $N_n(\boldsymbol{X\beta}, \boldsymbol{R}\sigma^2)$, where $\boldsymbol{R}$ is a $n \times n$ correlation matrix with entry $i,j = \theta^{|j-i|}$. The ordinary least squares estimator of $\boldsymbol{\beta}$ is again given by $\widehat{\boldsymbol{\beta}} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}$ as derived in the previous section. The sample variance is equal to $Var(\widehat{\boldsymbol{\beta}}) = Var((\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}) = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T Var(\boldsymbol{y})((\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T)^T = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{R}\sigma^2\boldsymbol{X}(\boldsymbol{X}^T\boldsymbol{X})^{-1} = \sigma^2(\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{R}\boldsymbol{X}(\boldsymbol{X}^T\boldsymbol{X})^{-1}$.

Given data we believe may be reasonably modelled with an AR(1) process, we can use the arima function in R where for example daily values can be modelled with arima(data, xreg=seq(start,end,1), order=c(1,0,0)). The function will estimate the intercept, slope, θ and ζ^2 and produce standard errors. With frequent data this is likely to be the best option. With less data and some reasonable assumptions regarding σ and θ, we can use the least squares estimators and create confidence intervals using the same methods as previous.

`1Supplementary file 2: R Code

```
# load packages
library(ggplot2)
library(cowplot)


############# Generate baseline data
# Baseline power data of a sample of 100 individuals are simulated from a hypothetical
# population with mean 3000 Watts and SD 750 Watts. These values represent the true scores
# at time 0.

# Model parameters
PopMean = 3000
PopSd = 750
SampleN = 100
set.seed(123)

# Simulate true scores
TS0Power = rnorm(SampleN,PopMean,PopSd)

# Check distribution of true scores
quantile(TS0Power, seq(0,1,0.1))

# Now we generate observed scores

# Data 1: We generate daily observed scores for a pre-intervention period of two weeks
# with noise set with a measurement error standard deviation (Typical Error [TE]) of 200 Watts
TE = 200

PowerBaseline = matrix(NA, nrow=SampleN, ncol=14)
for(i in 1:14){
  PowerBaseline[,i]=TS0Power + rnorm(SampleN,0,TE)
}

# Data 2: We generate daily observed scores that are repeated four times each day.
# We assume that the inter-session TE is 160 Watts and the intra-session TE is 120 Watts.
# We then have that sqrt(160^2+120^2) = 200 Watts
TEInter = 160
TEIntra = 120

# We save this data as an array
PowerBaselineTrueDay = matrix(NA,nrow=SampleN,ncol=14)
PowerBaselineIntraInter = array(NA, dim=c(SampleN,14,4))
for(i in 1:14){
  PowerBaselineTrueDay[,i]=TS0Power + rnorm(SampleN,0,TEInter)
  for(j in 1:4){
    PowerBaselineIntraInter[,i,j] = PowerBaselineTrueDay[,i]+rnorm(SampleN,0,TEIntra)
  }}

############# Visualise data 1 at baseline
MedianBaseline = apply(PowerBaseline, 1,median)
MedianBaselineOrder = order(MedianBaseline)
BaselineData = c(NULL)
for(i in 1:SampleN){
  BaselineData = c(BaselineData, PowerBaseline[MedianBaselineOrder[i],])
}
PowerBaselineDF = data.frame(Values = BaselineData,
                             Individual = factor(rep(1:SampleN,each=14)))

ggplot(PowerBaselineDF, aes(y=Values, fill=Individual))+ geom_boxplot() +
  ylab("Power (Watts)") + theme_classic() +
  theme(legend.position="none",axis.title.x=element_blank(),
        axis.text.x=element_blank(),
        axis.ticks.x=element_blank())
```

```
# 1000 by 700

############ Visualise typical error calculations

# We now want to visualise typical error values that will be estimated when using
# 1) the SD from individual participants;
# 2) test retest from samples of different sizes.

# Case1: Plot of typical error estimated from SD of individual participant baseline data
# The typical error estimates will be different for different participants due to sampling error.
TESD = apply(PowerBaseline, 1,sd)
TESDDF = data.frame(Value=TESD)
n = 100
mean = mean(TESDDF$Value)
sd = sd(TESDDF$Value)
binwidth = 20

SDTEPlot = ggplot(TESDDF, aes(x = Value, mean = mean, sd = sd, binwidth = binwidth, n = n)) +
  theme_classic() +
  geom_histogram(binwidth = binwidth,
                 colour = "white", fill = "cornflowerblue", size = 0.1) +
  stat_function(fun = function(x) dnorm(x, mean = mean, sd = sd) * n * binwidth,
                color = "darkred", size = 1) +
  xlab("Typical error (Watts)") + ylab("Count") +
  geom_vline(xintercept = TE,linetype="dashed", size =1.5)
SDTEPlot

quantile(TESD, c(0,0.25,0.5,0.75,1))

# Case 2: Plot of typical error from test-retest data

# We can calculate TE from different pairwise combinations of days, e.g.
# day 1 v.s. day 2 or day 2 v.s. day 14 and so on. So we list the pairwise combinations.
Pairwise14 = combn(1:14,2)

# We can also calculate TE from different samples from our 100 participants.
# Here we calculate test-retest TE from first 10,20,30,...100 across
# all pairwise combinations
Pairwise14TE = matrix(NA,nrow=10,ncol=91)
for(i in 1:10){
  for(j in 1:91)
    Pairwise14TE[i,j] =sd(PowerBaseline[(1:seq(10,100,10)[i]),
                                        Pairwise14[2,j]] -
                          PowerBaseline[(1:seq(10,100,10)[i]),
                                        Pairwise14[1,j]])/sqrt(2)
}

BaselineTestRetestData = c(NULL)
for(i in 1:10){
  BaselineTestRetestData = c(BaselineTestRetestData,Pairwise14TE[i,])}

BaselineTestRetestDataDF = data.frame(Values = BaselineTestRetestData,
                                      Group = factor(rep(seq(10,100,10),each=91)))

TestRetestTEPlot = ggplot(BaselineTestRetestDataDF, aes(y=Values,x=Group, fill=Group))+ geom_boxplot() +
  ylab("Typical error (Watts)") + theme_classic() +
  scale_y_continuous(limits=c(100,300),breaks=seq(100,300,50))+
  theme(legend.position="none") + xlab("Number of participants") +
  geom_hline(yintercept = 200,linetype="dashed", size =1.5)
TestRetestTEPlot

# Combine Plots
plot_grid(SDTEPlot, TestRetestTEPlot,labels = c('A', 'B'), hjust=-1.5,label_size = 12)
# 1000 by 650
```

```
############ Test true score proportions
# Now that we have our typical error calculations we want to visualise and assess the
# true score confidence intervals that we generate.
# Here we generate 95% confidence intervals using case 1: sd of participant scores

# We assess the proportion of intervals that contain the true score based on
# centring our intervals around day 1 observed values
Day1SDDF = data.frame(True = TS0Power,
                      LB = PowerBaseline[,1]-1.96*TESD,
                      UB = PowerBaseline[,1]+1.96*TESD)
mean(Day1SDDF$LB<Day1SDDF$True&
       Day1SDDF$UB>Day1SDDF$True)
# 0.95

# We now create a plot of the intervals and true scores
Day1SDDFOrder = Day1SDDF[order(Day1SDDF$True),]
Day1SDDFOrder$X = 1:SampleN
Day1SDDFOrder$C = Day1SDDFOrder$LB<Day1SDDFOrder$True&
  Day1SDDFOrder$UB>Day1SDDFOrder$True

SDPlot = ggplot(Day1SDDFOrder, aes(X, True)) +
  geom_point() +
  geom_errorbar(aes(ymin = LB, ymax = UB,color=C)) +
  ylab("Power (Watts)") + theme_classic() +
  scale_y_continuous(breaks=seq(1000,5000,1000))+
  theme(legend.position="none",axis.title.x=element_blank(),
        axis.text.x=element_blank(),
        axis.ticks.x=element_blank(),
        axis.line.x=element_blank())
SDPlot

# We repeat a similar process using case 2: test-retest typical error estimates.
# Here we generate 95% confidence intervals using test-retest data with 20
# participants from observed data from day 1 and day 2.The intervals are
# then applied to day 3 scores. We use the adjusted multiplier from sample of 20
# which for 95%CI is 2.1
Day1TestRetestDF = data.frame(True = TS0Power,
                              LB = PowerBaseline[,3]-2.1*Pairwise14TE[2,1],
                              UB = PowerBaseline[,3]+2.1*Pairwise14TE[2,1])
mean(Day1TestRetestDF$LB<Day1TestRetestDF$True&
       Day1TestRetestDF$UB>Day1TestRetestDF$True)
# 0.91

# We now create a plot of the intervals and true scores
Day1TestRetestDFOrder = Day1TestRetestDF[order(Day1TestRetestDF$True),]
Day1TestRetestDFOrder$X = 1:SampleN
Day1TestRetestDFOrder$C = Day1TestRetestDFOrder$LB<Day1TestRetestDFOrder$True&
  Day1TestRetestDFOrder$UB>Day1TestRetestDFOrder$True

TestRetestPlot =ggplot(Day1TestRetestDFOrder, aes(X, True)) +
  geom_point() +
  geom_errorbar(aes(ymin = LB, ymax = UB,color=C)) +
  ylab("Power (Watts)") + theme_classic() +
  scale_y_continuous(breaks=seq(1000,5000,1000))+
  theme(legend.position="none",axis.title.x=element_blank(),
        axis.text.x=element_blank(),
        axis.ticks.x=element_blank(),
          axis.line.x=element_blank())
TestRetestPlot

# Combine plots
TestRetestPlotCombine =ggplot(Day1TestRetestDFOrder, aes(X, True)) +
  geom_point() +
  geom_errorbar(aes(ymin = LB, ymax = UB,color=C)) +
```

```
  ylab("Power (Watts)") + theme_classic() +
  scale_y_continuous(breaks=seq(1000,5000,1000))+
  theme(legend.position="none",axis.title.x=element_blank(),
        axis.text.x=element_blank(),
        axis.ticks.x=element_blank(),
        axis.line.x=element_blank(),
        axis.text.y=element_blank(),
        axis.ticks.y=element_blank(),
        axis.line.y=element_blank(),
        axis.title.y=element_blank())

plot_grid(SDPlot, TestRetestPlotCombine,labels = c('A', 'B'), hjust=-1.5,label_size = 12)
# 1000 by 650

##########################################################################################
##########################################################################################
##########################          ##########################################

############ Framework extension 1 – Mean of multiple measurements

# In this example we visualise how the typical error changes when we take the
# mean of different numbers of measurements.To illustrate this process we randomly
# select 20 individuals from our sample. We then visualise the true score confidence
# intervals and how these change when we take the mean of 1,4,7 and 14 measurements.
# For ease we assume that the TE of 200 Watts is known,

# Select 20 random individuals.
set.seed(123)
R20 = sample(1:SampleN,20,replace=FALSE)
TS0Power20 = TS0Power[R20]
PowerBaseline20 = PowerBaseline[R20,]

# Calculate the mean of 1,4,7,14 measurements
MeasurementA = c(1,4,7,14)
MeanPowerBaseline20 = matrix(NA,nrow=20,ncol=4)
MeanPowerBaseline20[,1]=PowerBaseline20[,1]
for(i in 2:4){
  MeanPowerBaseline20[,i]=apply(PowerBaseline20[,1:MeasurementA[i]],1,mean)}

# Organise the data frame so that the participants are ordered in increasing true score

TS0Power20Order = TS0Power20[order(TS0Power20)]
MeanPowerBaseline20Order = MeanPowerBaseline20[order(TS0Power20),]

# Calculate the intervals bounds using the typical error and adjustment based on using
# the mean of different number of data points.

MeanPowerBaseline20OrderDF = data.frame(X = c(seq(1,80,4),seq(2,80,4),
                                              seq(3,80,4),seq(4,80,4)),
                                        True =rep(TS0Power20Order,4),
                                        LB = c(MeanPowerBaseline20Order[,1]-1.96*200,
                                               MeanPowerBaseline20Order[,2]-1.96*200/sqrt(4),
                                               MeanPowerBaseline20Order[,3]-1.96*200/sqrt(7),
                                               MeanPowerBaseline20Order[,4]-1.96*200/sqrt(14)),
                                        UB = c(MeanPowerBaseline20Order[,1]+1.96*200,
                                               MeanPowerBaseline20Order[,2]+1.96*200/sqrt(4),
                                               MeanPowerBaseline20Order[,3]+1.96*200/sqrt(7),
                                               MeanPowerBaseline20Order[,4]+1.96*200/sqrt(14)),
                                        Group = c(rep(letters[1:20],4)))

MeanPowerBaseline20OrderDF$C = MeanPowerBaseline20OrderDF$LB<MeanPowerBaseline20OrderDF$True&
  MeanPowerBaseline20OrderDF$UB>MeanPowerBaseline20OrderDF$True
```

```
# Visualise the intervals across the different number of measurements
TrueScoreCIPlot = ggplot(MeanPowerBaseline20OrderDF, aes(X, True, group=Group)) +
  geom_point() +
  geom_errorbar(aes(ymin = LB, ymax = UB,color=C)) +
  ylab("Power (Watts)") + theme_classic() +
  scale_y_continuous(breaks=seq(1000,5000,1000))+
  theme(legend.position="none",axis.title.x=element_blank(),
        axis.text.x=element_blank(),
        axis.ticks.x=element_blank(),
        axis.line.x=element_blank())+
  annotate("text",x=1, y = 2000, label = "P1")+  annotate("text", x = 5, y = 2300, label = "P2")+
  annotate("text",x=9, y = 2600, label = "P3")+  annotate("text", x = 13, y = 2600, label = "P4")+
  annotate("text",x=17, y = 3000, label = "P5")+  annotate("text", x = 21, y = 3400, label = "P6")+
  annotate("text",x=25, y = 3400, label = "P7")+  annotate("text", x = 29, y = 3400, label = "P8")+
  annotate("text",x=33, y = 3600, label = "P9")+  annotate("text", x = 37, y = 3750, label = "P10")+
  annotate("text",x=41, y = 3850, label = "P11")+  annotate("text", x = 45, y = 4100, label = "P12")+
  annotate("text",x=49, y = 4100, label = "P13")+  annotate("text", x = 53, y = 4100, label = "P14")+
  annotate("text",x=57, y = 4300, label = "P15")+  annotate("text", x = 61, y = 4400, label = "P16")+
  annotate("text",x=65, y = 4600, label = "P17")+  annotate("text", x = 69, y = 4600, label = "P18")+
  annotate("text",x=73, y = 4600, label = "P19")+  annotate("text", x = 77, y = 5300, label = "P20")
TrueScoreCIPlot
############# Framework extension 1a Inclusion of hierarchical model with inter- and intra typical errors

# First we start with some calculations to show how the intra- and inter-session TE values influence the
# overall uncertainty in mean scores calculated across different combinations of inter and intra-sessions.

# Inter-day TE of 160 Watts and Intra-day TE of 120 Watts
sqrt(TEInter^2+TEIntra^2)
# 200

# Here we show the different TE values based on scenarios highlighted in the paper

# Scenario 1: Mean of four measurements on single day
Scenario1TE = sqrt(TEInter^2+(TEIntra^2)/4)
Scenario1TE
# 170.8801

# Scenario 2: Mean of single measurements on four days
Scenario2TE = sqrt((TEInter^2+TEIntra^2)/4)
Scenario2TE
# 100

# Scenario 3: grand mean of for measurements on four days
Scenario3TE = sqrt(((TEInter^2)/4)+((TEIntra^2)/(4*4)))
Scenario3TE
# 85.44004

# Now plot how the CI intervals change across the scenarios for the 20 participants
# We start with our 20 participants and a single measurement on day 1
HierarchicalBase = PowerBaselineIntraInter[R20,1,1]

# Scenario 1 we calculate the mean of the four measurements in day 1
Hierarchical1Matrix = matrix(NA,nrow=20,ncol=4)
for(i in 1:4){
  Hierarchical1Matrix[,i]=PowerBaselineIntraInter[R20,1,i]}
Hierarchical1Mean = apply(Hierarchical1Matrix,1,mean)

# Scenario 2 we calculate the mean of firs measurement day 1,2,3,4
Hierarchical2Matrix = matrix(NA,nrow=20,ncol=4)
for(i in 1:4){
  Hierarchical2Matrix[,i]=PowerBaselineIntraInter[R20,i,1]}
Hierarchical2Mean = apply(Hierarchical2Matrix,1,mean)

# Scenario 3 we calculate the grand mean of four measurement day 1,2,3,4
Hierarchical3MatrixIntra = matrix(NA,nrow=20,ncol=4)
```

```
Hierarchical3MatrixInter = matrix(NA,nrow=20,ncol=4)
for(i in 1:4){
  for(j in 1:4){
    Hierarchical3MatrixIntra[,j]=PowerBaselineIntraInter[R20,i,j]}
  Hierarchical3MatrixInter[,i] = apply(Hierarchical3MatrixIntra,1,mean)}
Hierarchical3Mean = apply(Hierarchical3MatrixInter,1,mean)

# Combine into one data frame
# Organise the data frame so that the participants are ordered in increasing true score

TS0Power20Order = TS0Power20[order(TS0Power20)]

HierarchicalBaseOrder=HierarchicalBase[order(TS0Power20)]
Hierarchical1MeanOrder=Hierarchical1Mean[order(TS0Power20)]
Hierarchical2MeanOrder=Hierarchical2Mean[order(TS0Power20)]
Hierarchical3MeanOrder=Hierarchical3Mean[order(TS0Power20)]

MeanHierarchical20OrderDF = data.frame(X = c(seq(1,80,4),seq(2,80,4),
                                     seq(3,80,4),seq(4,80,4)),
                                True =rep(TS0Power20Order,4),
                                LB = c(HierarchicalBaseOrder-1.96*200,
                                      Hierarchical1MeanOrder-1.96*Scenario1TE,
                                      Hierarchical2MeanOrder-1.96*Scenario2TE,
                                      Hierarchical3MeanOrder-1.96*Scenario3TE),
                                UB = c(HierarchicalBaseOrder+1.96*200,
                                      Hierarchical1MeanOrder+1.96*Scenario1TE,
                                      Hierarchical2MeanOrder+1.96*Scenario2TE,
                                      Hierarchical3MeanOrder+1.96*Scenario3TE),
                                Group = c(rep(letters[1:20],4)))

MeanHierarchical20OrderDF$C = MeanHierarchical20OrderDF$LB<MeanHierarchical20OrderDF$True&
  MeanHierarchical20OrderDF$UB>MeanHierarchical20OrderDF$True

TrueScoreHierarchicalCIPlot = ggplot(MeanHierarchical20OrderDF, aes(X, True, group=Group)) +
  geom_point() +
  geom_errorbar(aes(ymin = LB, ymax = UB,color=C)) +
  ylab("Power (Watts)") + theme_classic() +
  scale_y_continuous(breaks=seq(1000,5000,1000))+
  theme(legend.position="none",axis.title.x=element_blank(),
        axis.text.x=element_blank(),
        axis.ticks.x=element_blank(),
        axis.line.x=element_blank(),
        axis.text.y=element_blank(),
        axis.ticks.y=element_blank(),
        axis.line.y=element_blank(),
        axis.title.y=element_blank())+
  annotate("text",x=1, y = 1900, label = "P1")+  annotate("text", x = 5, y = 2300, label = "P2")+
  annotate("text",x=9, y = 2400, label = "P3")+  annotate("text", x = 13, y = 2700, label = "P4")+
  annotate("text",x=17, y = 3400, label = "P5")+  annotate("text", x = 21, y = 3400, label = "P6")+
  annotate("text",x=25, y = 3400, label = "P7")+  annotate("text", x = 29, y = 3400, label = "P8")+
  annotate("text",x=33, y = 3800, label = "P9")+  annotate("text", x = 37, y = 3800, label = "P10")+
  annotate("text",x=41, y = 3800, label = "P11")+  annotate("text", x = 45, y = 4200, label = "P12")+
  annotate("text",x=49, y = 4200, label = "P13")+  annotate("text", x = 53, y = 4200, label = "P14")+
  annotate("text",x=57, y = 4300, label = "P15")+  annotate("text", x = 61, y = 4300, label = "P16")+
  annotate("text",x=65, y = 4600, label = "P17")+  annotate("text", x = 69, y = 4600, label = "P18")+
  annotate("text",x=73, y = 4600, label = "P19")+  annotate("text", x = 77, y = 5100, label = "P20")
TrueScoreHierarchicalCIPlot

# Combined plots
plot_grid(TrueScoreCIPlot, TrueScoreHierarchicalCIPlot,labels = c('A', 'B'), hjust=-1.5,label_size = 12)
# 1400 by 700

#######  We now run a simple check on true score CIs using known errors
# We conclude this section by showing that CIs using the methods outlined produce the
# correct proportions (e.g. 0.95) when the TEs are known.
```

```
set.seed(123)
PopN = 10000
Ts = rnorm(PopN,PopMean,PopSd)

# Scenario 1
ObsSingle4 = matrix(NA, nrow = PopN, ncol=4)
Day1S1 = Ts+ rnorm(PopN, 0, TEInter)
for(i in 1:4){
  ObsSingle4[,i] = Day1S1+ rnorm(PopN,0,TEIntra)}
ObsSingle4mean = apply(ObsSingle4,1,mean)
round(mean(ObsSingle4mean -1.96*Scenario1TE<Ts&ObsSingle4mean+1.96*Scenario1TE>Ts),2)
# 0.95

# Scenario 2
Daily4 = matrix(NA, nrow = PopN, ncol=4)
ObsDaily4 = matrix(NA, nrow = PopN, ncol=4)
for(i in 1:4){
  Daily4[,i]=Ts + rnorm(PopN, 0, TEInter)
  ObsDaily4[,i]=Daily4[,i] + rnorm(PopN, 0,TEIntra)}
ObsDaily4mean = apply(ObsDaily4,1,mean)
round(mean(ObsDaily4mean -1.96*Scenario2TE<Ts&ObsDaily4mean+1.96*Scenario2TE>Ts),2)
# 0.95

# Scenario 3
ObsFour4 = array(NA, dim=c(PopN,4,4))
for(i in 1:4){
  for(j in 1:4){
    ObsFour4[,j,i] = Daily4[,i]+rnorm(PopN,0,TEIntra)}}
ObsFour4mean = matrix(NA, nrow = PopN, ncol=4)
for(i in 1:4){
  ObsFour4mean[,i] = apply(ObsFour4[,,i],1,mean)}
ObsFour4Gmean = apply(ObsFour4mean,1,mean)
round(mean(ObsFour4Gmean -1.96*Scenario3TE<Ts&ObsFour4Gmean+1.96*Scenario3TE>Ts),2)
# 0.95


################################################################################
#######
################################################################################
#######
############################          ##########################################

############# Framework extension 2 – Intermediate testing

# Here we use Monte Carlo simulation to conceptualise capturing uncertainty with
# intermediate testing.

# Create plot illustrating example from paper

# Set up participant true scores across ten weeks assuming 30 Watt improvement
# per week for 10 weeks, with measurements taken every 2 weeks and a TE of 200 Watts
IntermediateBase = 3000
IntermediateDuration = 10
IntermediateFrequency = 2
IntermediateGradient = 30
IntermediateExampleTrue = seq(IntermediateBase,
                              IntermediateBase+(IntermediateDuration*IntermediateGradient),
                              IntermediateFrequency*IntermediateGradient)
IntermediateN = length(IntermediateExampleTrue)
IntermediateTE = 200
IntermediateIterations = 10
# Create observed scores
set.seed(1)
IntermediateExampleObs = IntermediateExampleTrue + rnorm(IntermediateN,0,IntermediateTE)
```

```
IntermediateExampleDraws = rep(IntermediateExampleObs,IntermediateIterations)+
                    rnorm(IntermediateN*IntermediateIterations,0,IntermediateTE)

IntermediateExampleDF = data.frame(Y = c(IntermediateExampleObs,IntermediateExampleDraws),
                           X = rep(seq(0,IntermediateDuration,IntermediateFrequency),IntermediateIterations+1),
                           group = factor(c(rep("Obs",IntermediateN),
                                    rep("MC",IntermediateN*IntermediateIterations))),
                           group2 = factor(rep((1:(IntermediateIterations+1)),each=IntermediateN)))

# Fit regression lines
IntermediateExampleRM  = matrix(NA, nrow = 2, ncol =IntermediateIterations+1)
RSeq = seq(1,length(IntermediateExampleDF[,1]),IntermediateN)
for(i in 1:(IntermediateIterations+1)){
   LM = lm(Y~X,data=IntermediateExampleDF[(RSeq[i]:(RSeq[i]+(IntermediateN-1))),])
   IntermediateExampleRM[1,i]=summary(LM)$coefficient[1,1]
   IntermediateExampleRM[2,i]=summary(LM)$coefficient[2,1]}
IntermediateExampleRM

# 1st plot of single regression line from observed data
SingleRegressionPlot = ggplot(IntermediateExampleDF,aes(x=X,y=Y,color=group))+geom_point() +
 scale_color_manual(values=c("black", "red")) +
 geom_abline(intercept = IntermediateExampleRM[1,1],
          slope = IntermediateExampleRM[2,1],color="red")+
 scale_x_continuous(breaks = seq(0,10,1))+
 theme_classic()+ xlab("Week") + ylab("Power (Watts)") +
 theme(legend.position="none")
SingleRegressionPlot

# 2nd plot with regression lines from random sample based on first set of
# observed data
AllRegressionPlot = ggplot(IntermediateExampleDF,aes(x=X,y=Y,color=group2))+geom_point() +
 scale_color_manual(values=c("red", "black","slategray","springgreen",
                           "steelblue2","tan","turquoise","yellow",
                           "gold3","blue4","grey20")) + theme_classic()+
 geom_abline(intercept = IntermediateExampleRM[1,1],
          slope = IntermediateExampleRM[2,1],color="red",alpha=0.4)+
 geom_abline(intercept = IntermediateExampleRM[1,2],
          slope = IntermediateExampleRM[2,2],color="black",alpha=0.4)+
 geom_abline(intercept = IntermediateExampleRM[1,3],
          slope = IntermediateExampleRM[2,3],color="slategray",alpha=0.4)+
 geom_abline(intercept = IntermediateExampleRM[1,4],
          slope = IntermediateExampleRM[2,4],color="springgreen",alpha=0.4)+
 geom_abline(intercept = IntermediateExampleRM[1,5],
          slope = IntermediateExampleRM[2,5],color="steelblue2",alpha=0.4)+
 geom_abline(intercept = IntermediateExampleRM[1,6],
          slope = IntermediateExampleRM[2,6],color="tan",alpha=0.4)+
 geom_abline(intercept = IntermediateExampleRM[1,7],
          slope = IntermediateExampleRM[2,7],color="turquoise",alpha=0.4)+
 geom_abline(intercept = IntermediateExampleRM[1,8],
          slope = IntermediateExampleRM[2,8],color="yellow",alpha=0.4)+
 geom_abline(intercept = IntermediateExampleRM[1,9],
          slope = IntermediateExampleRM[2,9],color="gold3",alpha=0.4)+
 geom_abline(intercept = IntermediateExampleRM[1,10],
          slope = IntermediateExampleRM[2,10],color="blue4",alpha=0.4)+
 geom_abline(intercept = IntermediateExampleRM[1,11],
          slope = IntermediateExampleRM[2,11],color="grey20",alpha=0.4)+
 scale_x_continuous(breaks = seq(0,10,1))+
xlab("Week") + ylab("Power (Watts)") +
 theme(legend.position="none")

AllRegressionPlot

# Using the small sample of regression lines quantify uncertainty in improvement
(10*IntermediateExampleRM[2,])[order(IntermediateExampleRM[2,])]
quantile(10*IntermediateExampleRM[2,],c(0.025,0.25,0.5,0.75,0.975))
```

```
# Combine Plots
plot_grid(SingleRegressionPlot, AllRegressionPlot,labels = c('A', 'B'), ncol = 1)
# 1000 by 600


# Development of true score change confidence intervals using large number of Monte Carlo Iterations
set.seed(1234)
IntermediateMCRegression = c(NULL)
for(i in 1:10000){
  Data = data.frame(X =seq(0,10,2),
                    Y = IntermediateExampleObs + rnorm(6,0,IntermediateTE))
  IntermediateMCRegression[i] = 10*summary(lm(Y~X,data=Data))$coefficient[2,1]}

quantile(IntermediateMCRegression,c(0.025,0.25,0.5,0.75,0.975))

#75% CI
quantile(IntermediateMCRegression,c(0.125,0.875))


# Least squares regression example.
# We recreate the above Monte Carlo simulation analytically using least squares

X = matrix(c(rep(1,6),seq(0,10,2)),nrow=6,ncol=2)
y = matrix(c(2874,3097,2953,3499,3305,3136),nrow=6,ncol=1)

Betahat = solve(t(X)%*%X)%*%t(X)%*%y
SampleVar = IntermediateTE^2*solve(t(X)%*%X)

# Approximate 75%CI
10*(Betahat[2,1]-1.15*sqrt(SampleVar[2,2]))
10*(Betahat[2,1]+1.15*sqrt(SampleVar[2,2]))

# Plot showing true score change intervals across different measurement frequencies
# for Monte Carlo and least squares regression

# Function to create data for 100 realisations of observed scores across different
# measurement frequencies
Data100 = function(IntermediateFreq){
  True = seq(IntermediateBase,
             IntermediateBase+(IntermediateDuration*IntermediateGradient),
             IntermediateFreq*IntermediateGradient)
  N = length(True)
  DataCollect = matrix(NA, nrow=100,ncol=N)
  for(i in 1:100){
    DataCollect[i,]=True + rnorm(N,0,IntermediateTE)}
  return(DataCollect)}

# Function to run Monte Carlo Simulation method on data and calculate CIs of given %
MonteCarloCI = function(Data,timepoints,ProportionCI){
  N = length(timepoints)
  CIBounds = matrix(NA,nrow=100,ncol=2)
  for(i in 1:100){
    ChangeEstimate = c(NULL)
    for(j in 1:1000){
      DataI = data.frame(X =timepoints,
                         Y = Data[i,] + rnorm(N,0,IntermediateTE))
      ChangeEstimate[j] = 10*summary(lm(Y~X,data=DataI))$coefficient[2,1]}
    CIBounds[i,1]=quantile(ChangeEstimate,c(((1-ProportionCI)/2),1-((1-ProportionCI)/2)))[[1]]
    CIBounds[i,2]=quantile(ChangeEstimate,c(((1-ProportionCI)/2),1-((1-ProportionCI)/2)))[[2]]}
  return(CIBounds)}

# Function to run analytical method on data and calculate CIs of given %
LeastSquaresCI = function(Data,timepoints,ProportionCI){
  # Calculate multiplier
```

```
 Multiplier =  -1*qnorm((1-ProportionCI)/2)
X = matrix(c(rep(1,length(Data[1,])),timepoints),nrow=length(Data[1,]),ncol=2)
LSE = c(NULL)
SE = sqrt((IntermediateTE^2*solve(t(X)%*%X))[2,2])
CIBounds = matrix(NA,nrow=100,ncol=2)
for(i in 1:100){
  y = as.matrix(Data[i,])
  LSE[i]=(solve(t(X)%*%X)%*%t(X)%*%y)[2,1]
  CIBounds[i,1]=10*(LSE[i]-(Multiplier*SE))
  CIBounds[i,2]=10*(LSE[i]+(Multiplier*SE))}
return(CIBounds)}


# Three time points
set.seed(123)
Data3 = Data100(5)
MonteCarlo3T95 = MonteCarloCI(Data=Data3,timepoints=c(0,5,10),ProportionCI=0.95)
mean(MonteCarlo3T95[,1]<300&MonteCarlo3T95[,2]>300)
LeastSquares3T95 = LeastSquaresCI(Data=Data3,timepoints=c(0,5,10),ProportionCI=0.95)
mean(LeastSquares3T95[,1]<300&LeastSquares3T95[,2]>300)

# Six time points
set.seed(123)
Data6 = Data100(2)
MonteCarlo6T95 = MonteCarloCI(Data=Data6,timepoints=seq(0,10,2),ProportionCI=0.95)
mean(MonteCarlo6T95[,1]<300&MonteCarlo6T95[,2]>300)
LeastSquares6T95 = LeastSquaresCI(Data=Data6,timepoints=seq(0,10,2),ProportionCI=0.95)
mean(LeastSquares6T95[,1]<300&LeastSquares6T95[,2]>300)

# Eleven time points
set.seed(12)
Data11 = Data100(1)
MonteCarlo11T95 = MonteCarloCI(Data=Data11,timepoints=seq(0,10,1),ProportionCI=0.95)
mean(MonteCarlo11T95[,1]<300&MonteCarlo11T95[,2]>300)
LeastSquares11T95 = LeastSquaresCI(Data=Data11,timepoints=seq(0,10,1),ProportionCI=0.95)
mean(LeastSquares11T95[,1]<300&LeastSquares11T95[,2]>300)

# Twenty-one time points
set.seed(12)
Data21 = Data100(0.5)
MonteCarlo21T95 = MonteCarloCI(Data=Data21,timepoints=seq(0,10,0.5),ProportionCI=0.95)
mean(MonteCarlo21T95[,1]<300&MonteCarlo21T95[,2]>300)
LeastSquares21T95 = LeastSquaresCI(Data=Data21,timepoints=seq(0,10,0.5),ProportionCI=0.95)
mean(LeastSquares21T95[,1]<300&LeastSquares21T95[,2]>300)

# Plot
MC3DF = data.frame(LB = MonteCarlo3T95[,1],UB=MonteCarlo3T95[,2],
                   C=MonteCarlo3T95[,1]<300&MonteCarlo3T95[,2]>300,
                   X=seq(331,430,1))
MC3DF$Mid = (MC3DF$UB + MC3DF$LB)/2

MC6DF = data.frame(LB = MonteCarlo6T95[,1],UB=MonteCarlo6T95[,2],
                   C=MonteCarlo6T95[,1]<300&MonteCarlo6T95[,2]>300,
                   X=seq(221,320,1))
MC6DF$Mid = (MC6DF$UB + MC6DF$LB)/2

MC36DF = rbind(MC3DF,MC6DF)

MC11DF = data.frame(LB = MonteCarlo11T95[,1],UB=MonteCarlo11T95[,2],
                   C=MonteCarlo11T95[,1]<300&MonteCarlo11T95[,2]>300,
                   X=seq(111,210,1))
MC11DF$Mid = (MC11DF$UB + MC11DF$LB)/2

MC3611DF = rbind(MC3DF,MC6DF,MC11DF)
```

```
MC21DF = data.frame(LB = MonteCarlo21T95[,1],UB=MonteCarlo21T95[,2],
                C=MonteCarlo21T95[,1]<300&MonteCarlo21T95[,2]>300,
                X=seq(1,100,1))
MC21DF$Mid = (MC21DF$UB + MC21DF$LB)/2

MC361121DF = rbind(MC3DF,MC6DF,MC11DF,MC21DF)

MCPlot = ggplot(MC361121DF, aes(X, Mid, group=C)) +
  geom_point(alpha=0) +
  geom_errorbar(aes(ymin = LB, ymax = UB,color=C))+
  ylab("Power (Watts)") + theme_classic() +
  scale_y_continuous(breaks=seq(-1000,1500,200))+
  coord_flip()+
  theme(legend.position="none",axis.title.y=element_blank(),
        axis.text.y=element_blank(),
        axis.ticks.y=element_blank()) +
  geom_hline(yintercept = 300,linetype="dashed", size =0.8)

LS3DF = data.frame(LB = LeastSquares3T95[,1],UB=LeastSquares3T95[,2],
                C=LeastSquares3T95[,1]<300&LeastSquares3T95[,2]>300,
                X=seq(331,430,1))
LS3DF$Mid = (LS3DF$UB + LS3DF$LB)/2

LS6DF = data.frame(LB = LeastSquares6T95[,1],UB=LeastSquares6T95[,2],
                C=LeastSquares6T95[,1]<300&LeastSquares6T95[,2]>300,
                X=seq(221,320,1))
LS6DF$Mid = (LS6DF$UB + LS6DF$LB)/2

LS36DF = rbind(LS3DF,LS6DF)

LS11DF = data.frame(LB = LeastSquares11T95[,1],UB=LeastSquares11T95[,2],
                C=LeastSquares11T95[,1]<300&LeastSquares11T95[,2]>300,
                X=seq(111,210,1))
LS11DF$Mid = (LS11DF$UB + LS11DF$LB)/2

LS3611DF = rbind(LS3DF,LS6DF,LS11DF)

LS21DF = data.frame(LB = LeastSquares21T95[,1],UB=LeastSquares21T95[,2],
                C=LeastSquares21T95[,1]<300&LeastSquares21T95[,2]>300,
                X=seq(1,100,1))
LS21DF$Mid = (LS21DF$UB + LS21DF$LB)/2

LS361121DF = rbind(LS3DF,LS6DF,LS11DF,LS21DF)

LSPlot = ggplot(LS361121DF, aes(X, Mid, group=C)) +
  geom_point(alpha=0) +
  geom_errorbar(aes(ymin = LB, ymax = UB,color=C))+
  ylab("Power (Watts)") + theme_classic() +
  scale_y_continuous(breaks=seq(-1000,1500,200))+
  coord_flip()+
  theme(legend.position="none",axis.title.y=element_blank(),
        axis.text.y=element_blank(),
        axis.ticks.y=element_blank(),
        axis.line.y=element_blank()) +
  geom_hline(yintercept = 300,linetype="dashed", size =0.8)

plot_grid(MCPlot, LSPlot,labels = c('A', 'B'), hjust=-1.5,label_size = 12)
# 1000 by 600

##########################################################################
#######
##########################################################################
#######
##########################          ##########################################
```

```
############ Autoregressive Modelling

# We assume that a single measurement is made each day and we model errors from
# AR(1) models with autocorrelation of 0.2 and 0.8, and in both cases sigma equal to 200.

# Build AR(1)
# for theta 0.2 we have var(y)= 200^2 = zeta^2 / 1-0.2^2; hence zeta = 196
# for theta 0.2 we have var(y)= 200^2 = zeta^2 / 1-0.8^2; hence zeta = 120

set.seed(112)
UPower0.2 = WPower0.2 =rnorm(71,0,196)
for(i in 2:71){
  UPower0.2[i] = 0.2*UPower0.2[i-1] + WPower0.2[i]}
sd(UPower0.2)

set.seed(112)
UPower0.8 = WPower0.8 =rnorm(71,0,120)
for(i in 2:71){
  UPower0.8[i] = 0.8*UPower0.8[i-1] + WPower0.8[i]}
sd(UPower0.8)

# Add linear regression to AR(1) Errors
Power0.2 = c(NULL)
for(i in 1:71){
  Power0.2[i]=3000 + ((i-1)*(300/70)) + UPower0.2[i]
}

Power0.8 = c(NULL)
for(i in 1:71){
  Power0.8[i]=3000 + ((i-1)*(300/70)) + UPower0.8[i]
}

# Plot data
PowerIDF = data.frame(Obs0.2 = Power0.2,
                      Obs0.8 = Power0.8,
                      Day = seq(0,70,1),
                      True = seq(3000,3300,(300/70)))

AR0.2Plot = ggplot(PowerIDF, aes(x=Day,Obs0.2)) +
  geom_point() + theme_classic() +
  geom_abline(intercept = 3000, slope = 300/70,color="red")+
  scale_x_continuous(limit = c(0,70), breaks = seq(0,70,5))+
  scale_y_continuous(limit = c(2500,3750),breaks = seq(2500,3750,250))+
  theme_classic()+ xlab("Day") + ylab("Power (Watts)") +
  geom_segment(aes(x=0:70,y=c(rep(0,14),PowerIDF$Obs0.2[15:43],
                              rep(0,28)),xend=1:71,
                   yend=c(rep(0,14),PowerIDF$Obs0.2[16:44],
                          rep(0,28))))


AR0.2Plot

AR0.8Plot = ggplot(PowerIDF, aes(x=Day,Obs0.8)) +
  geom_point() + theme_classic() +
  geom_abline(intercept = 3000, slope = 300/70,color="red")+
  scale_x_continuous(limit = c(0,70), breaks = seq(0,70,5))+
  scale_y_continuous(limit = c(2500,3750),breaks = seq(2500,3750,250))+
  theme_classic()+ xlab("Day") + ylab("Power (Watts)") +
  geom_segment(aes(x=0:70,y=c(rep(0,14),PowerIDF$Obs0.8[15:54],
                              rep(0,17)),xend=1:71,
                   yend=c(rep(0,14),PowerIDF$Obs0.8[16:55],
                          rep(0,17))))

AR0.8Plot
```

```
# Combine plots
plot_grid(AR0.2Plot,AR0.8Plot,labels = c('A', 'B'), hjust=-1.5,label_size = 12)
# 1000 by 600

# Fit AR(1) models using the arima function in R and least squares

# Function that returns the least squares estimate of the regression slope and
# standard error. Assumes serial correlation and TE are known.

LeastSquaresAR1 = function(Data,timepoints,rho){
  n =length(timepoints)
  # Create design matrix
  X = matrix(c(rep(1,length(timepoints)),timepoints),nrow=length(timepoints),ncol=2)
  # Create correlation matrix
  exponent = abs(matrix(1:n - 1, nrow = n, ncol = n, byrow = TRUE) -
                   (1:n - 1))
  R = rho^exponent
  LSE = (solve(t(X)%*%X)%*%t(X)%*%Data)[2,1]
  SE = sqrt((IntermediateTE^2*(solve(t(X)%*%X)%*%t(X)%*%R%*%X%*%solve(t(X)%*%X)))[2,2])
  return(c(LSE,SE))}

# arima calculation 0.2
arima0.2 = arima(PowerIDF$Obs0.2, xreg=seq(0,70,1), order=c(1,0,0))
arima0.2
# Estimated improvement across intervention
arima0.2$coef[[3]]*70
# 95% CI
70*(arima0.2$coef[[3]]-1.96*sqrt(arima0.2$var.coef[[3,3]]))
70*(arima0.2$coef[[3]]+1.96*sqrt(arima0.2$var.coef[[3,3]]))

# arima calculation 0.8
arima0.8 = arima(PowerIDF$Obs0.8, xreg=seq(0,70,1), order=c(1,0,0))
arima0.8
# Estimated improvement across intervention
arima0.8$coef[[3]]*70
# 95% CI
70*(arima0.8$coef[[3]]-1.96*sqrt(arima0.8$var.coef[[3,3]]))
70*(arima0.8$coef[[3]]+1.96*sqrt(arima0.8$var.coef[[3,3]]))

#least squares calculation
# calculation 0.2
LSAR10.2 = LeastSquaresAR1(PowerIDF$Obs0.2,seq(0,70,1),0.2)
# Estimated improvement across intervention
LSAR10.2[1]*70
# 95% CI
70*(LSAR10.2[1]-1.96*LSAR10.2[2])
70*(LSAR10.2[1]+1.96*LSAR10.2[2])

# calculation 0.8
LSAR10.8 = LeastSquaresAR1(PowerIDF$Obs0.8,seq(0,70,1),0.8)
# Estimated improvement across intervention
LSAR10.8[1]*70
# 95% CI
70*(LSAR10.8[1]-1.96*LSAR10.8[2])
70*(LSAR10.8[1]+1.96*LSAR10.8[2])
```