CARLETON UNIVERSITY

SCHOOL OF
MATHEMATICS AND STATISTICS

HONOURS PROJECT

TITLE: Resampling Diagnostics: Bootstrap, Shiny, and Democratizing Data Science

AUTHOR: Alexander Lehmann

SUPERVISOR: Dr. Dave Campbell

DATE: August 20, 2021

Resampling Diagnostics
Bootstrap, Shiny, and Democratizing Data Science

Alexander Lehmann

August 20, 2021

Preface

In the summer of 2015, I spent an enlightening period employed as a mechanical engineering intern at a scientific instrument manufacturing company in the forests of my native New Hampshire. Early in my term I was made responsible for maintaining and operating the company's 3D printer, a venerable machine which demanded constant monitoring as it ran and a quasi-religious ritual of maintenance procedures and occasional cusswords between printing runs. Operating this abomination of poorly-molded plastic soon consumed a significant portion of my time, as the demand for its product quickly outstripped the man-hours available for the machine's maintenance, setup, and monitoring. My other tasks, much more aligned with the role of a mechanical engineer, fell by the wayside as nearly every working hour was consumed by this device.

Eventually, though, the company decided that an upgrade was required and my mechanical nemesis was replaced by a gleaming new Ultimaker 2 professional-grade 3D printer. The machine took what was previously a task defined by careful attention and finger-crossing and turned it into a fire-and-forget operation requiring no real intervention. The formerly-laborious setup and tweaking process was replaced by automated software tools handling the minutiae of the printing parameters. Producing a part changed from a painstaking process only I could undertake to one that any other employee could complete, even without the dedicated schooling and 3D printer experience that the older machine had demanded.

As the burden of the 3D printing tasks were taken off of my shoulders and distributed across the entire company, I was suddenly able to devote my time to my other work. With the 3D printing problems out of the way, I was able to complete other projects that doubled production capacity for a popular product, provided diagnostic tools for new designs, and identified manufacturing defects in outsourced parts. The provision of a single tool allowed advancements in many areas unrelated at once, purely by making what was previously a task requiring specific, expert knowledge accessible to those without it.

Around the same time, the company's marketing department was shifting to a data-driven approach to their work. Data was collected, reports tabulated, and analysis conducted. However, the marketing department were not the ones conducting these analyses; while motivated and hardworking, they simply lacked the specific expertise to accomplish the task. Instead, the analytical work was shifted to the software engineering staff as the closest employees to data scientists at the company. This, naturally, took away time from software engineering work and led to inefficiencies much as 3D printing tasks had led to inefficiencies for me.

As I approach the end of my undergraduate study in statistics, I wonder if well-designed tools could have resolved this inefficiency in the way a well-designed tool resolved my own. Could a carefully-crafted software suite have allowed the marketers to complete their analyses internally, sparing the consumption of engineering resources? Could such software tools provide an advantage over simply hiring a statistician or data scientist? Might the deep domain knowledge of the marketing team lead to better insights than the greater technical skills of the engineers, given the right tools?

These questions are the foundation of what is being called the *democratization* of data science [3]; the concept that tools which are user-friendly and methodologically robust can allow even people without specific training in statistics to ask questions of data and obtain reliable answers. Surely it is better for a company's staff to each be able to complete the basic analyses required for their functions while the more knowledgeable statisticians and data scientists are freed to engage with more complex and difficult problems. We already see some of these tools available: consider Google Analytics, which automatically collects and visualizes web traffic data, or the plethora of COVID-19 dashboards published throughout 2020, some of which integrate regression modeling and other such tools.

With this concept in mind, I set out to develop a tool to expand the accessibility of the bootstrap with democratic data principles. The bootstrap is a widely-applicable and tremendously useful tool which unlocks analysis of many models, such as nonparametric and nonlinear models, where traditional statistical inference is difficult. While this project was initially supposed to be a simple exploration of diagnostic methods for the bootstrap, it quickly expanded into a study of nonparametric modeling, robust statistics, various other resampling methods, and software engineering. It is my hope that the product of my labors can serve as an example of the feasibility and utility of democratic principles as the data science field continues to evolve.

1 An Overview of Bootstrap Methods

The bootstrap is a resampling method whereby we approximate the distribution of a statistic by computing it many times on data sets resampled with replacement from some population sample. While computationally intensive, it requires little in the way of complicated formulae and generally does not rely on any assumptions about the distribution of a statistic. These advantages allow the bootstrap to estimate distributions of and assign measures of accuracy to statistics for which closed-form solutions are difficult to compute or do not exist at all. Common applications of the bootstrap include estimating standard error, bias, confidence intervals, and distribution features such as skewness and kurtosis.

1.1 Case Resampling

We begin with the simplest version of the bootstrap, the one-sample nonparametric situation. This situation frequently arises with inference for simple statistics such as the sample median. Let $\mathbf{X} = (X_1, X_2, \ldots, X_n)$ be a random sample drawn from some distribution F and let the random variable $R(\mathbf{X}, F)$ be some statistic of interest whose distribution we wish to know. The simplest bootstrap procedure for this situation is the case-resampling procedure, for which we present Efron's [4] original description:

1. Construct the sample probability distribution $\hat{F}$, putting mass $1/n$ at each point $x_1, x_2, x_3, \ldots, x_n$.
2. With $\hat{F}$ fixed, draw a random sample of size n from $\hat{F}$, say

$$X_i^* = x_i^* \overset{iid}{\sim} \hat{F}, i = 1, 2, \ldots, n \tag{1}$$

   Call this the *bootstrap sample*, $\mathbf{X}^* = (X_1^*, X_2^*, \ldots, X_n^*)$, $\mathbf{x}^* = (x_1^*, x_2^*, \ldots, x_n^*)$. Notice that we are not getting a permutation distribution since the values of $\mathbf{X}^*$ are selected *with* replacement from the set $\{x_1, x_2, \ldots, x_n\}$.
3. Approximate the sampling distribution of $R(\mathbf{X}, F)$ by the *bootstrap distribution* of

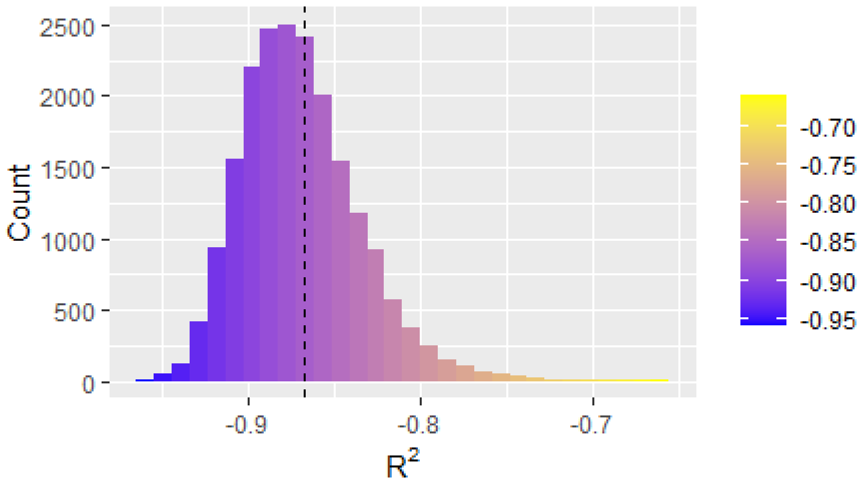
$$R^* = R(\mathbf{X}^*, \hat{F}) \tag{2}$$

   i.e., the distribution of R^* induced by the random mechanism (1), with $\hat{F}$ held fixed at its observed value.

Essentially, this procedure in the one-sample nonparametric situation approximates the distribution of the population with empirical distribution of the random sample $\mathbf{X}$ and uses this empirical distribution as the probability model from which the bootstrap samples are drawn. This substitution allows the simulation of synthetic datasets on which to compute $R(\mathbf{X}^*, \hat{F})$ as in (2).

Example 1. The Motor Trend Car Road Tests data set [10], 'mtcars' in the R language, contains design and performance details for 32 vehicles from the 1973-1974 model year. Suppose we are interested in the standard error of the correlation between vehicle weight ('wt' in R) and fuel efficiency ('mpg'). We compute the correlation coefficient of the two features and obtain a point estimate of -0.8677. Since we are not assured of the features' distributions we cannot estimate its distribution by traditional methods; instead, we turn to the bootstrap.

Figure 1: Histogram of bootstrap replications for Corr(wt, mpg)



We draw $B = 20,000$ bootstrap samples from the data with replacement and compute the correlation coefficient for vehicle weight and fuel efficiency in each. The standard error of these bootstrap estimates is 0.0345, which serves as a reasonable approximation for the standard error of the statistic itself.

1.2 Semi-Parametric Bootstrap

There are certain models which lend themselves to alternate probability models for resampling. Consider the problem of estimating the distribution of parameters for a given regression model. In this situation, we assume that the data follows some specified data-generating process with some amount of random noise. This regression assumption provides for an alternate (and often more accurate) resampling model by which we resample from the model residuals and use these to generate our bootstrap data, rather than resampling from the original sample itself. We call this method the *semi-parametric*, or *residual*, bootstrap.

The procedure begins with the correct specification of some regression model, which may be parametric or nonparametric. We the proceed as follows:

1. Fit the regression model to the sample data $\mathbf{X}$ and y.

2. Compute the model's predicted responses

$$\hat{y} = \{\hat{y}_i | i = 1, 2, \ldots, n\} \tag{3}$$

at each x_i in the sample and the residuals

$$\hat{\epsilon_i} = \{\hat{y}_i - y_i | i = 1, 2, \ldots, n\}. \tag{4}$$

3. Construct the sample residual probability distribution $\hat{F}$, putting mass $1/n$ at each residual $\epsilon_1, \epsilon_2, \ldots, \epsilon_n$.

4. With $\hat{F}$ fixed, draw a random sample of size n from $\hat{F}$, say

$$E_i^* = \epsilon_i^* \overset{iid}{\sim} \hat{F}, i = 1, 2, \ldots, n. \tag{5}$$

Calculate the bootstrap responses

$$y_i^* = \hat{y}_i + \epsilon_i^*, i = 1, 2, \ldots, n. \tag{6}$$

5. Calculate the regression parameters for covariates $\mathbf{X}$ and responses $\hat{y}_i$.

6. Approximate the sampling distributions of the parameters in the same manner as in the nonparametric case-resampling scheme.

By including the residuals and predicted responses in the probability model, we achieve a resampling scheme that more closely mirrors the assumed data-generating process for the data. This allows more accurate inference using the bootstrap and improves estimates.

Example 2.

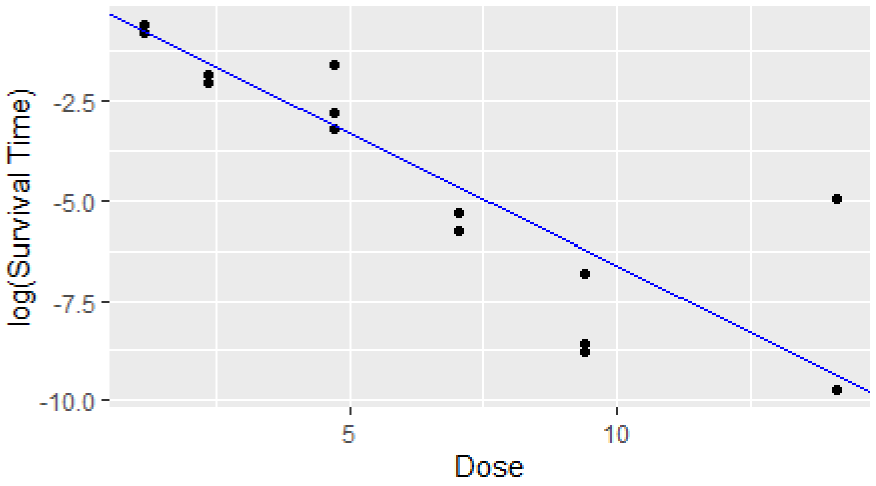
The cell survival data [7] contains the results of an experiment in which a radiologist exposed 14 bacterial plates to different doses of radiation and recorded the proportion surviving the bombardment (Figure 2). Suppose we wish to construct a 95 percent confidence interval for the slope of the regression line using the bootstrap. We first fit a linear regression model to the data, seen in Figure 2. This regression fit appears to model the data-generating process well, so we may use the semi-parametric bootstrap to construct a confidence interval for the slope.

We begin by computing the predicted survival times at each radiation dosage using the fitted model. We next draw $B = 1000$ samples with replacement from the residual values of the model fit and add these residuals to the predicted values in a component-wise fashion to produce our 1000 bootstrap data sets. For each of these data sets, we fit a new linear regression model and record the slope.

From these bootstrap replications, we construct the confidence interval using the percentile method, where the $1 - 2\alpha$ confidence interval is calculated as

$$\left[\hat{\theta}_{\%,\mathrm{lo}}, \hat{\theta}_{\%,\mathrm{hi}}\right] \approx \left[\hat{\theta}_B^{*(\alpha)}, \hat{\theta}_B^{*(\alpha)}\right]. \tag{7}$$

Figure 2: Plot of log(survival time) vs radiation dose with OLS line



This produces the interval $(-0.7739, -0.5578)$ (Figure 3). The result is comparable to the interval $(-0.7870, -0.5397)$ computed by traditional methods, but required no distributional assumptions on the residuals.

Remark. *Other methods of calculating bootstrap confidence intervals exist (*$\mathrm{BC_A}, ABC, \ldots$*), which may give more accurate confidence bounds or be more efficient estimators. We chose to use the percentile method for its simplicity and ease of use, which we felt appropriate given our focus on democratizing data science.*

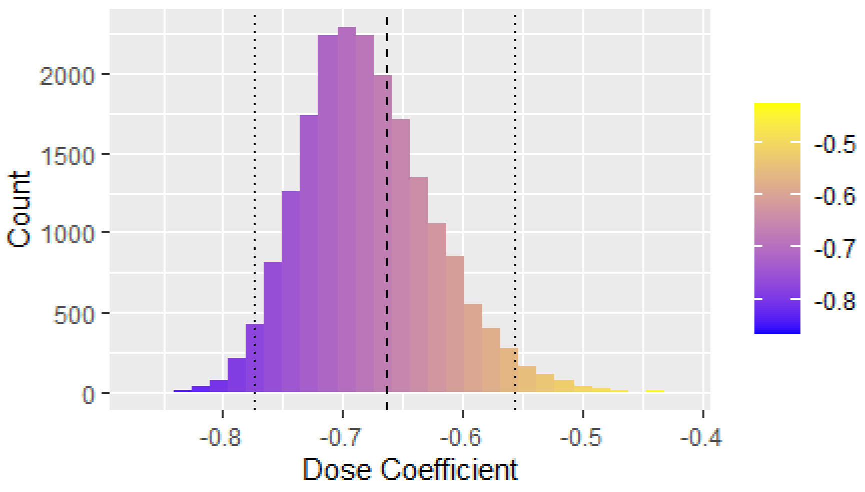
1.3 Stratified Resampling

Recall that the bootstrap proceeds by analogy; the bootstrap replication is to the original estimate as the original sample is to the population. This analogy relies upon the assumption that the bootstrap samples are similar to and representative of sampling from the original distribution. This "resample like sample" principle can easily break down when data is not completely homogeneous, such as when the data include samples from multiple populations, such as different species or other categorical variations.

Example 3. The ‘iris‘ data set [8] contains 150 observations of morphological features of iris flowers belonging to three species: *iris setosa*, *iris versicolor*, and *iris virginica*. The data set contains 50 observations for each species, proportionally $\{\frac{1}{3}, \frac{1}{3}, \frac{1}{3}\}$ of the total data.

Suppose we use the case-resampling bootstrap to generate $B = 5$ bootstrap samples (this is an extremely small number of bootstrap samples for any practical purpose, but suffices to illustrate the principle at work here). We see

Figure 3: Histogram of bootstrap replications for OLS line slope



that each of the bootstrap samples has a different distribution of species within it, which may lead to unrepresentative bootstrap replications of a statistic or model. These unrepresentative replications can lead to inconsistency in the bootstrap and misleading approximations of statistic distributions, impeding robust methodology.

In a situation such as this, alternative sampling schemes may be used to rectify the class imbalance in the bootstrap samples and generate better resampling models. One such scheme is the stratified resampling scheme for the bootstrap. In this scheme, we ensure that each bootstrap sample contains the same balance of class labels as the original by sampling from *within* each class, then aggregating these subsamples to form the bootstrap samples. One method for stratified resampling is seen below.

1. Select a categorical stratifying variable S in the data. For each of the levels of S, say the ith, count the number of cases in the sample with class label S_i. Call the count of cases b_i.

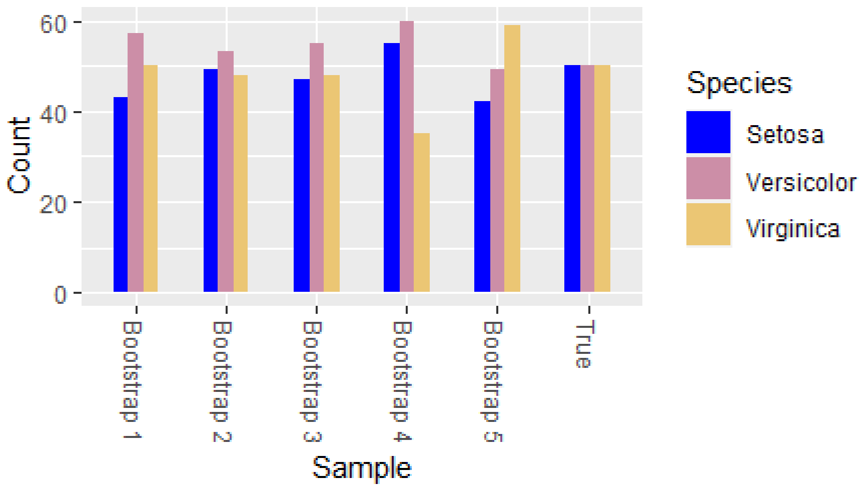
2. For each level of S, say the ith, resample b_i times with replacement from the subset of sample observations with class label S_i. Combine each of these subsamples to form the bootstrap sample.

3. Proceed with inference using the bootstrap samples as normal.

2 The Jackknife and Jackknife-After-Bootstrap

The bootstrap methods discussed thus far provide a variety of ways to select a bootstrap sample, though none of them include any form of diagnostics to

Figure 4: Plot of Cases by Species in Bootstrap Samples from ‘iris’ Data



check if the bootstrap samples are representative of a population sample. The jackknife-after-bootstrap technique provides a method for checking the influence of points in the bootstrap sample on the approximated sampling distribution and for detecting outliers that may require further examination.

2.1 The Jackknife

The jackknife [12] is an alternative resampling scheme to the bootstrap for estimating variance and bias of a statistic. While the bootstrap is built upon random sampling with replacement, the jackknife systematically leaves out one of the n observations in the data set to obtain n subsamples of size $n - 1$. Formulae exist to estimate the statistic, its standard error, and its variance from these subsamples.

As with the bootstrap, the jackknife does not require parametric assumptions regarding the distribution of the statistic of interest nor of the distribution of the sample (i.e. we do not have to assume either follow some parametric distribution). However, it can still require significant computational resources for large data sets and provides only an approximation of the statistic and its distributional properties.

A standard procedure for jackknife estimation of a summary statistic $s(\mathbf{X})$ proceeds as follows:

1. Collect a sample $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ of size n from a chosen population.
2. Generate n subsamples $\mathbf{x}_{(i)}, i = 1, \ldots, n$ of $\mathbf{x}$ where

$$\mathbf{x}_{(i)} = (x_1, x_2, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n) \tag{8}$$

   is the subsample with case i deleted.

3. Calculate $s_{(i)}(\mathbf{x}) = s(\mathbf{x}_{(i)})$, the ith case-deletion estimate of $s(\mathbf{X})$, for each subsample generated.

4. The jackknife estimate of the statistic is the mean of the case-deletion estimates, that is,
$$\hat{s}(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n} s_{(i)}(\mathbf{x}). \tag{9}$$

5. The jackknife estimate of the variance of the statistic is the variance of the case-deletion estimates, that is,
$$\widehat{\text{Var}}(s(\mathbf{x})) = \frac{n-1}{n}\sum_{i=1}^{n}(s_{(i)}(\mathbf{x}) - \hat{s}(\mathbf{x}))^2. \tag{10}$$

6. The jackknife estimate of bias of the statistic is the difference between the jackknife estimate and the full-data estimate scaled for the sample size, that is,
$$\widehat{\text{Bias}}(s(\mathbf{x})) = (n-1)(\hat{s}(\mathbf{x}) - s(\mathbf{x})). \tag{11}$$

Example 1. We again consider the Motor Trend Car Road Tests data set of Section 1.1. Suppose we are interested in the correlation between the vehicles' motor power, measured in horsepower (hp), and fuel economy measured in miles per gallon (mpg). We will use the jackknife to find an estimate of the correlation and estimates of its variance and bias.

There are $n = 32$ cars in the data set, so we create 32 case-deletion subsamples. For illustration, consider the subsample created by deleting the $i = 23$rd car,
$$\mathbf{x}_{(23)} = (x_1, x_2, \ldots, x_{22}, x_{24}, \ldots, x_{32}). \tag{12}$$
We compute the correlation between the motor power and fuel economy within this subset, obtaining a case-deletion estimate of $\text{Corr}_{(23)}(\text{hp}, \text{mpg}) \approx -0.7835$.

We repeat this process for each of the 31 other cases, obtaining 32 case-deletion estimates $\text{Corr}_{(\text{i})}(\text{hp}, \text{mpg}), i = 1, \ldots, 32$. The jackknife estimate of the correlation between motor power and fuel economy is the mean of these case-deletion estimates,

$$\widehat{\text{Corr}}(\text{hp}, \text{mpg}) = \frac{1}{32}\sum_{i=1}^{32} \text{Corr}_{(\text{i})}(\text{hp}, \text{mpg}) \approx -0.7766. \tag{13}$$

We can now calculate the jackknife estimate of the variance:

$$\widehat{\text{Var}}(\text{Corr}(\text{hp}, \text{mpg})) = \frac{n-1}{n}\sum_{i=1}^{32}\left(\text{Corr}_{(\text{i})}(\text{hp}, \text{mpg}) + 0.7766\right)^2 \approx 0.0010. \tag{14}$$

To calculate the jackknife estimate of the bias, we use the full data set to compute an estimate for the correlation between motor power and fuel economy, $r(\text{hp}, \text{mpg}) \approx -0.7762$. The jackknife estimate of the bias is then

$$\widehat{\text{Bias}}(\text{Corr}(\text{hp}, \text{mpg})) = (31)(-0.7766 + 0.7762) = -0.0124. \tag{15}$$

2.2 Jackknife Influence Functions

A key use of the jackknife is to identify those data which have a significant disruptive effect on the statistic of interest. The jackknife's leave-one-out subsampling creates a convenient environment within which to assess the impact of each individual datum.

The primary tool used to assess such impacts in the jackknife world is the *jackknife influence function*, defined below.

Definition. Suppose that $s(\mathbf{x})$ is a real-valued statistic of interested. Let $\mathbf{x}_{(i)}$ indicate the data set remaining after deletion of the ith point,

$$\mathbf{x}_{(i)} = (x_1, x_2, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n), \tag{16}$$

and let $s_{(i)} = s(\mathbf{x}_{(i)})$, the corresponding deleted point value of the statistic of interest. The *jackknife influence function* for s is defined to be

$$u_i(s) = (n-1)(\overline{s_{()}} - s_{(i)}) \tag{17}$$

where $\overline{s_{()}} = \sum_{i=1}^{n} s_{(i)}/n$.

The jackknife influence function quantifies the effect a given datum has on the estimate of the statistic of interest. The scaling factor accounts for the size of the data set; for a fixed estimate difference $\overline{s_{()}} - s_{(i)}$, a large data set size n will result in a greater influence function value since the deleted case must compete with more data.

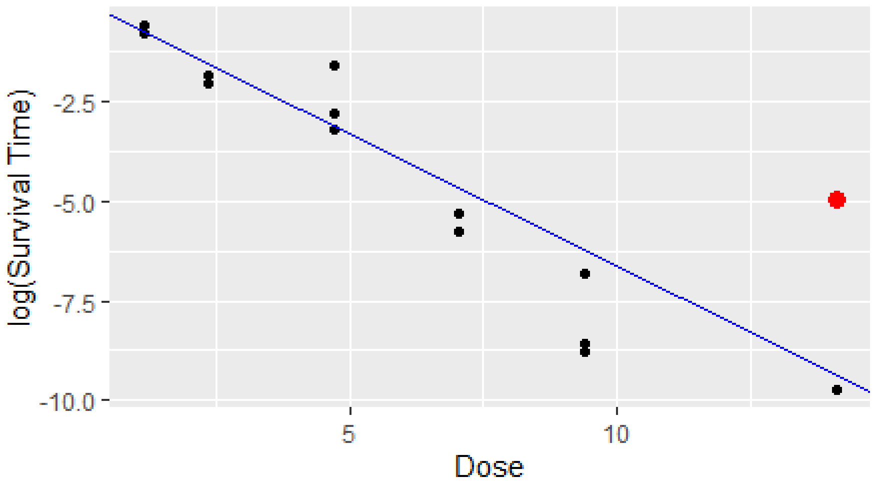
Example 2.

In this example, we return to the cell survival data from Section 1.2. Note that one particular observation, case 13 which is highlighted in red, appears to lie far from the regression line. This case may have a disruptive effect on the regression slope, which we would like to quantify by computing the jackknife influence value for this case.

We jackknife the OLS linear regression slope, obtaining $n = 14$ case-deletion estimates of the slope. The mean of these case deletion estimates is

$$\overline{\beta_{()}} = \sum_{i=1}^{n} \beta_{(i)}/n \approx -0.6643. \tag{18}$$

Figure 5: Plot of log(survival time) vs radiation dose with OLS line; case 13 highlighted



The case-deletion estimate computed from the subsample leaving out case 13 is $\beta_{(13)} \approx -0.7596$, so the jackknife influence value for case 13 is

$$u_{13}(\beta) = (13)(-0.6643 + 0.7596) = 1.2389. \tag{19}$$

Compare the case 13 influence value to the influence values for some other cases, such as $u_2(\beta) \approx -0.0088$ and $u_5(\beta) = -0.0196$.

Clearly, the jackknife influence function's value is directly affected by the dispersion of the data; in order to account for this, we often prefer to consider an adjusted form.

Definition. Let $u_i(s)$ be the jackknife influence function for a given case i and statistic $s(\mathbf{X})$. The *relative jackknife influence function* is

$$u_i^\dagger(s) = u_i(s) \Bigg/ \left[\sum_j \frac{u_j(s)^2}{n-1}\right]^{1/2}. \tag{20}$$

The relative jackknife influence function prevents the dispersion of the data from affecting the dispersion of the jackknife influence, simplifying the assessment of individual cases' impacts on the estimated value. The relative influence value provides an objective measurement, independent of features of distributional assumptions, of the impact each datum has on the estimated value of the statistic of interest and thus serves as an indication of the robustness of the estimator.

2.3 The Jackknife-After-Bootstrap Sampling Lemma

The jackknife procedure described so far requires modification to be practically useful for diagnostic purposes in the bootstrap. If $s(\mathbf{X})$ above is taken to be some bootstrap statistic, then each case-deletion estimate $s_{(i)}(\mathbf{X})$ will require a new set of B bootstrap samples to be drawn. For large data set sizes and/or values of B, this can quickly become computationally prohibitive and prevent jackknife-after-bootstrap from being conducted.

This problem may be circumvented via the *jackknife-after-bootstrap sampling lemma* [7]:

Lemma 2.1. *A bootstrap sample drawn with replacement from* $x_1, x_2, \ldots x_{i-1}, x_{i+1}, \ldots x_n$ *has the same distribution as a bootstrap sample drawn from* $x_1, x_2, \ldots x_n$ *in which none of the bootstrap values equals* x_i*.*

The jackknife-after-bootstrap sampling lemma allows the analyst to create jackknife subsamples of the bootstrap replications by simply including all of the bootstrap replications calculated from bootstrap samples in which a particular case does not appear. The lemma assures us that the distribution of these selected bootstrap replications is the same as it would be if we calculated a new set of replications from $\mathbf{x}_{(i)}$. This shortcut makes jackknife-after-bootstrap analysis feasible with limited computational resources and enables its use in practice, as it requires no further computation at all after the initial set of bootstrap samples have been drawn.

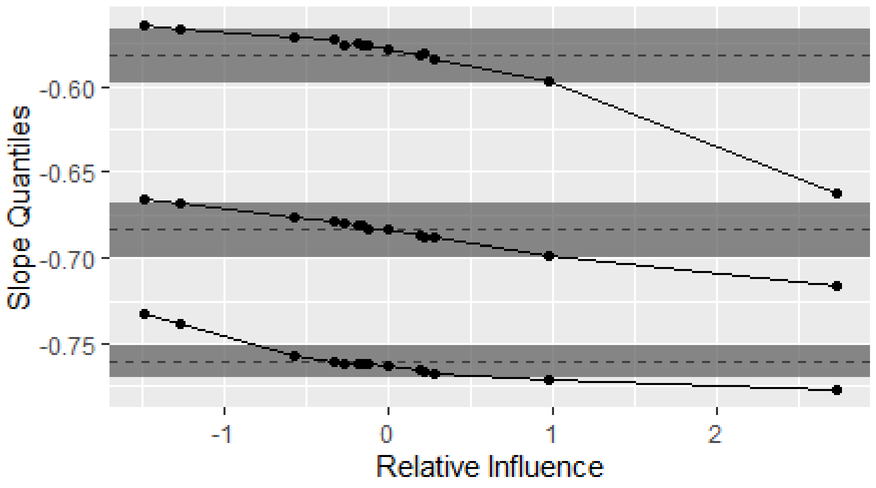
2.4 Outlier Detection

Naturally, the investigation of individual cases' influence on the estimates of a statistic of interest lead to questions about outliers in the data. Comparison between cases' disruptive effects on statistic estimation and sampling distribution approximation may highlight those cases which may be removed for better bootstrap results. Canty, Davison, Hinkley & Ventura [1] describe a graphical method by which the analyst may highlight the disruptive effect that an outlier may have on bootstrap calculations.

Their method uses the *jackknife-after-bootstrap plot*, constructed by plotting chosen quantiles of case-deletion estimates $s_{(i)}(\mathbf{X})$ against the case's relative influence value $u_i^{\dagger}(s)$; that is, we plot the case's disruptive effect on the bootstrap distribution against its disruptive effect on the estimate. The points for each quantile are commonly connected for clarity. It is also common to include horizontal dashed lines corresponding to the same quantiles of the full-data bootstrap replications. Their described addition to the traditional plot is an uncertainty band around each of these full-data quantile lines where the width of the band is $2 \times z$ times the standard deviation estimate based on the interquartile range of the case-deletion quantiles and $z = 1.96$ based on normal theory (Canty, Davison, Hinkley & Ventura 2006). Case-deletion quantiles lying outside of this band may be considered outliers and investigated, modified, or deleted as the analyst deems appropriate.

Example 3.

Figure 6: Jackknife-after-bootstrap plot for cell survival data; 0.05, 0.50, and 0.95 quantiles plotted



Suppose that we now wish to search for outliers affecting the estimate of OLS regression slope for the cell survival data. We use the semi-parametric bootstrap to produce $B = 20,000$ bootstrap samples and fit the regression model as in Example 2.

We next use the jackknife-after-bootstrap method to obtain $n = 14$ case-deletion estimates for the regression slope. From these case-deletion estimates we calculate the jackknife influences as in Example 2. Then we find the 5%, 50%, and 95% percentiles for the full-data bootstrap replications and the jackknife-after-bootstrap data sets, and compute the uncertainty bands for each quantile.

The jackknife-after-bootstrap plot is constructed by plotting the case-deletion estimate quantiles for the slope against the relative influence of each point. The horizontal broken lines correspond to the full-data bootstrap quantiles and the shaded area corresponds to each quantile's calculated uncertainty band as described by Canty et al. We see that on the right, case 13's quantiles lie far outside each uncertainty band. This indicates that case 13 has a severe disruptive effect on the bootstrap calculations, suggesting that the analyst should review this case. We also see two other cases that may warrant additional review, although their effects do not appear to be as severe as case 13.

2.5 Extension to Multivariate Statistics

One shortcoming of the outlier detection mechanism already described is its incapacity to directly handle vector statistics. For some models such as multiple linear regression, this shortcoming precludes using jackknife-after-bootstrap plots for outlier detection without modification to the procedure. We propose an extension by which the difference calculations in the formulae are instead replaced by the L_2, or Euclidean, norm of the difference between the two vectors. This replacement allows the calculation of jackknife influence values and outlier detection in spaces with more than a single dimension.

Definition. Suppose that $\mathbf{s}(\mathbf{X}) = (s_1, s_2, \ldots, s_p), s_1, s_2, \ldots, s_p \in \mathbb{R}$ is a statistic of interest. Let $x_{(i)}$ indicate the data set remaining after deletion of the ith point,

$$\mathbf{x}_{(i)} = (x_1, x_2, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n), \tag{21}$$

and let $\mathbf{s}_{(i)} = \mathbf{s}(\mathbf{x}_{(i)})$, the corresponding deleted point value of the statistic of interest. The *multivariate jackknife influence function* for $\mathbf{s}$ is defined to be

$$v_i(\mathbf{s}) = (n-1)\|\overline{\mathbf{s}_{()}} - \mathbf{s}_{(i)}\|_2 \tag{22}$$

where $\overline{\mathbf{s}_{()}} = \left(\sum_{i=1}^n s_{1_{(i)}}/n, \sum_{i=1}^n s_{2_{(i)}}/n, \ldots, \sum_{i=1}^n s_{p_{(i)}}/n\right)$ is the vector of jackknife estimates for each component of $\mathbf{s}$.

Remark. *Note that for 1-dimensional statistics of interest $s(\mathbf{X})$, the multivariate jackknife influence function is equivalent to the absolute value of the single-dimensional jackknife influence function.*

As in the unmodified procedure, it is often useful to have a relative measure of influence; the calculation of relative multivariate jackknife influence is identical to the relative jackknife influence.

Definition. Let $v_i(\mathbf{s})$ be the multivariate jackknife influence function for a given case i and statistic $\mathbf{s}(\mathbf{X})$. The *relative multivariate jackknife influence function* is

$$v_i^\dagger(\mathbf{s}) = v_i(\mathbf{s}) \Big/ \left[\sum_j \frac{v_i(\mathbf{s})^2}{n-1}\right]^{1/2}. \tag{23}$$

We additionally need a method by which to assess bootstrap distribution disruption.

Definition. Suppose we have used the bootstrap to estimate the distribution of some vector statistic $\mathbf{s}(\mathbf{X}) = (s_1, s_2, \ldots, s_p)$, obtained bootstrap replications $\mathbf{s}_i^* = (s_{1_i}^*, s_{2_i}^*, \ldots, s_{p_i}^*), i = 1, \ldots, B$, and used jackknife-after-bootstrap to obtain case-deletion estimates $\mathbf{s}_{(i)}, i = 1, \ldots, n$ for each case. Let

$$J_i = \{s_{i_{(1)}}, s_{i_{(2)}}, \ldots, s_{i_{(n)}}\} \tag{24}$$

be the set of case-deletion estimates of the ith component of statistic $\mathbf{s}$ and

$$K_i = \{s_{i_1}^*, s_{i_2}^*, \ldots, s_{i_B}^*\} \tag{25}$$

be the set of bootstrap replications of the ith component of statistic $\mathbf{s}$. Let $Q_{\mathbf{x}}(q)$ be the sample quantile function for $\mathbf{x}$ and let

$$\delta_{\mathbf{s}}(q) = (Q_{J_1}(q) - Q_{K_1}(q), Q_{J_2}(q) - Q_{K_2}(q), \ldots, Q_{J_p}(q) - Q_{K_p}(q)) \qquad (26)$$

be the vector of differences between the qth sample quantiles of J and K for each component, i.e. the vector of differences between the qth quantile of the case-deletion estimates and the qth quantile of the full-data bootstrap replications. Then we define the *bootstrap disruption distance* for quantile q as the sum of squares of the components of $\delta_{\mathbf{s}}(q)$,

$$d_{\mathbf{s}}(q) = \sum_{i=1}^{p} (Q_{J_i} - Q_{K_i})^2. \qquad (27)$$

While similar to the modification made for the jackknife influence, we use the sum of squares of the differences rather than the L_2 norm. This is to allow a natural extension of the normal theory-based uncertainty bands in the single-variable situation to the multivariate case here. Since the procedure for each component individually is the same as in the single-variable situation, an appropriate uncertainty band for the individual components would be the 0.95 quantile of the standard normal distribution. Accordingly, the appropriate uncertainty band width for the sum of squares of these components is the 0.95 quantile of the chi-square distribution with p degrees of freedom, where p is the dimension of the vector statistic.

Remark. *Distributions exist for the use of the L_2 norm in this case (namely the chi distribution), but they are quite obscure compared to the chi-square distribution's relative ubiquity. As such, we have opted to use the sum of squares and chi-square distribution.*

3 Turner

With our foundational discussion of the bootstrap and jackknife-after-bootstrap complete, we turn our attention to the software we have developed to implement these techniques. Turner is a web app built on the R Shiny platform to ”black box” bootstrap procedures and jackknife-after-bootstrap, automating resampling model selection and diagnostics. Several models are supported and more can easily be added thanks to a modular design, and additional diagnostics are also possible. The extant version of Turner (at time of writing) is merely a prototype for possible future development, intended to serve as a proof-of-concept for such software and evince its usefulness in industry and academia.

The current release of Turner may be accessed online through any major browser. Full source code may be found on GitHub.

3.1 A Brief Overview of R Shiny

The R language [13] is a domain-specific programming language for statistical computing and data analysis. Its open-source licensing strategy allows users to develop and share *packages* to extend the language's functionality; some examples include the glmnet package to add LASSO and elastic-net regression models, R Markdown for generating reproducible reports, and parallel to enable parallel processing.

One of these packages is Shiny [2], developed and maintained by RStudio. Shiny extends R's functionality to include developing web apps running R code, allowing R programmers to easily create GUIs for users to interact with data and analyses. The purposes of these apps vary from interactive data visualization to modeling to diagnostic and IT support functions. Some examples of Shiny apps include RStudio's movie explorer, California's COVID Assessment Tool, and a simulation of COVID-19 vaccination strategies in Ontario.

A Shiny app has two parts: the UI and the server. The UI defines the controls displayed to and interacted with by the user, which can be divided into *input controls* and *output controls*. Input controls are parts of the UI which allow the user to modify the processes running within the app. These could be fields to change numeric values, sliders to change the value of a variable, or upload fields for data files. Output controls are parts of the UI which return data to the user, such as plots or download buttons.

The link between the inputs and outputs of a Shiny app is the server, which tends to be the more complex part of the app. The server is the part of the web app that actually performs the computations specified by the programmer and provides outputs to the UI to display to the user. A key feature of Shiny app servers is the *reactive programming model*, which only performs computations for parts of the app affected by the user's recent input. This model minimizes the amount of computation the app performs at any given time, making Shiny apps fast and efficient.

3.2 Bootstrap Automation

Turner uses R and Shiny to automate bootstrapping procedures by selecting the model, resampling, and performing diagnostics without requiring any particular expertise in these areas from the user. This automation is the key feature of Turner, allowing even users without training in bootstrap methods per se to employ these tools to produce usable results.

Model selection is performed by a combination of designer specification and statistical hypothesis testing, blending foreknowledge of the app's supported models with analysis of provided data. Turner is designed with modularity in mind, which applies in this instance by encapsulating each model's bootstrapping procedure(s) and evaluation procedure in its own file. For many of the statistics and models supported at time of writing, only a single bootstrap procedure is necessary. For those where multiple models may be sensible, it is trivial (from a programming standpoint) to add logic for selecting the procedure to be used. Stratification is dependent on the data provided and user's intent, so stratified resampling procedures are activated by user election through the UI. See Section 3.3 - Supported Models and Statistics for details.

The bootstrap procedures are executed using the rsample package [14], published as part of RStudio's tidymodels [11]. It provides a lightweight interface through which bootstrap resampling may be performed, reducing memory overhead and improving performance. This also has the benefit of standardizing the interface between the model-specific bootstrap procedures and the actual bootstrapping, simplifying the programmer's workflow to add new models and statistics. However, rsample does not support resampling models beyond case resampling with stratification; as a result, we must program our own methods for evaluating more complex schemes such as the semi-parametric bootstrap.

Fortunately, jackknife-after-bootstrap computations do not require consideration of alternate schemes or other complicating factors. This procedure may be applied without consideration to any factors apart from the bootstrap samples and replications, and thus required only that we program the jackknife-after-bootstrap procedure into the app. The results of this procedure are displayed as an interactive jackknife-after-bootstrap plot, along with a list of cases which lie outside the uncertainty bands computed using our modified procedure; this list serves to highlight cases which the analyst may wish to review for correction or removal, depending on circumstances external to Turner's purpose.

3.3 Supported Models and Statistics

At time of writing, Turner supports six models and statistics: the mean, median, correlation, linear regression, smoothing splines, and LOESS. Together, these serve as examples of the capabilities of Turner, and by extension the bootstrap, applied to several statistics and regression models both parametric and nonparametric. We note that some of these, particularly the mean and linear regression, do not necessarily require the bootstrap to estimate measures of spread or bias. These were included as validation for the various procedures and to demonstrate the reliability of Turner for statistical inference.

We now present a very brief overview of each statistic and model with details regarding their corresponding bootstrap procedures in Turner.

3.3.1 Mean

For a random sample $\mathbf{X} = (X_1, X_2, \ldots, X_n)$ of size n drawn from a probability distribution F, the sample mean $\overline{\mathbf{X}}$ serves as an estimate of central tendency for the probability distribution F. The common arithmetic mean is computed as

$$\overline{\mathbf{X}} = \sum_{i=1}^{n} \frac{X_i}{n}. \tag{28}$$

The mean is a simple statistic without residual concerns or other complicating matters for bootstrap inference, so we use the case-resampling bootstrap for bootstrapping the mean. Depending on the data from which the sample is drawn, the user may wish to modify the bootstrap procedure by introducing stratified resampling.

3.3.2 Median

The median is an alternative measure of central tendency, using the "middle" value of a set of numbers as an estimate of the center of the population. Formally, we define median($\mathbf{X}$) to be the value of the set $\mathbf{X}$ such that no more than half of the other observations are above or below median($\mathbf{X}$).

The median, like the mean, is a simple statistic and requires no special accommodations to perform bootstrapping. As such, we use case resampling with the potential for a stratified sampling modification as required.

3.3.3 Correlation

The correlation (or dependence) $\rho(X, Y)$ is a common measure of the relationship between two random variables X and Y. In particular, a large correlation indicates a strong linear predictive relationship between the two variables. For a random sample draw from a bivariate population $(X_1, Y_1), (X_2, Y_2), \ldots, (X_n, Y_n)$, we use the sample correlation coefficient

$$r_{xy} = \frac{\sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \overline{x})^2 \sum_{i=1}^{n}(y_i - \overline{y})^2}}. \tag{29}$$

The Turner bootstrap procedure for the correlation is, again, the case-resampling method. However, in this instance we resample paired observations rather than single observations, adjusting for stratifying variables as required.

3.3.4 Linear Regression

Regression modeling is a set of techniques for explaining and/or predicting the behaviour of one or more response variables as a function of one or more covariate variables. Linear regression is a subset of these techniques in which the statistician assumes that the relationship between the response and predictors is of the form

$$\mathbf{y} = X\beta + \epsilon \tag{30}$$

where $\mathbf{y}$ is a vector of observed values of the response, X is a matrix of row vectors encoding the corresponding observations of the covariates, β is a vector of regression coefficients in the model, and ϵ is a vector of random errors, also termed the residuals. Linear regression models typically require several assumptions, such as the model errors be uncorrelated.

Once we have specified a linear model for some data, we must fit the model to the data. This involves using some method to estimate the regression coefficients β in the true data-generating process by using the observed data. Turner currently supports two methods of linear model fitting: ordinary least squares (OLS) and iteratively re-weighted least squares (IRWLS) using an M-estimator. OLS fits assume that the model errors have independent and identical normal distributions with mean $\mu = 0$ and variance $\sigma^2 < \infty$ and attempt to minimize the mean squared error between the model and the observed data. IRWLS fits do not require such assumptions and is more robust, but requires greater computational resources to complete.

Bootstrapping linear regression models ordinarily that the analyst select between the case-resampling bootstrap and the semi-parametric bootstrap based on their own judgement and expertise. However, Turner automates this procedure by considering the distribution of the residuals to select the most appropriate resampling model, particularly whether or not they are identically distributed. When the user selects a linear model in the procedure setup, Turner performs a White test of heteroskedasticity on the data. If the residuals are homoskedastic, residual resampling is appropriate and this method is used. If not, the case-resampling method is used.

3.3.5 Smoothing Splines

Cubic splines are a method of approximating a function by using a piecewise function of cubic functions connected at a number of knots. In particular, a cubic spline requires that

1. The component cubic functions of the spline have equal value, first derivative, and second derivative at each knot;

2. The second derivative of the spline is zero at the endpoints.

This guarantees that the spline is a smooth interpolation of the data, but the statistician still must solve the problem of determining the number and location of the knots.

One solution to this problem is to sidestep it entirely by selecting a maximal set of knots; in particular, we designate each of the cases in the observed sample to be a knot. In this case, the complexity of the spline is handled by the use of a smoothing parameter $\lambda \in (0, \infty)$ which penalizes the residual sum of squares

$$\text{RSS}(f, \lambda) = \sum_{i=1}^{n} \{y_i - f(x_i)\}^2 + \lambda \int \{f''(t)\}^2 dt. \tag{31}$$

A more curved spline will incur a larger penalty term, regularizing the fit to prevent overfitting. A cubic spline specified in this manner is known as a smoothing spline. The smoothing parameter can be determined automatically and is the only parameter to the model, so λ is the statistic of interest for bootstrapping purposes.

While smoothing splines may share some superficial similarities to regression modeling by interpolating observations based on observed relationships, there are no residuals or other similar components to employ semi-parametric bootstrapping. Accordingly, we approximate the sampling distribution of the estimated smoothing parameter by case resampling.

3.3.6 LOESS

Local regression is class of nonparametric regression methods which use a smooth distance-based weight function, called a kernel, to assign weights to the observed data. These weights are then used to fit a regression model for a given locale in the data before moving along and repeating the procedure. These local regression models' estimates then form the overall regression model for the data.

LOESS is one of the most common methods of local regression. It uses a quadratic least-squares regression model to construct local estimates based on the tri-cube kernel

$$K_\lambda(x) = \begin{cases} \left(1 - |d|^3\right)^3, & d \leq \lambda \\ 0, & d > \lambda \end{cases} \tag{32}$$

where d is the distance between the currently-estimated part of the curve and the point x and λ is a smoothing parameter that specifies the width of the window under consideration. Conveniently, the LOESS fit requires only the specification of the smoothing parameter λ; in a sense, the supplied data *is* the model.

Unlike interpolating splines, LOESS does have residuals that could be resampled from to derive bootstrap approximations for the curve. However, we could not find an example in literature of semi-parametric bootstrap applied to LOESS fits and felt that (potentially) pioneering such an employment was beyond the scope of this paper. For this reason and in accordance with Turner's emphasis on simplicity, we opted to use the case-resampling bootstrap for the LOESS model.

3.3.7 Additional Models

Turner's modular design also makes it easy to implement additional models and statistics. New definitions require only an estimation and a bootstrap evaluation method and allow additional internal methods as desired. For example, consider the definition for the mean:

```
# Estimation ############################################
estimate_mean <- function(df, spec){
  mean(pull(df, as.name(spec$var)))
}

# Bootstrap evaluation ##################################
eval_mean <- function(df, spec) {
  mutate(
    df,
    replication = map_dbl(
                        sample,
                        estimate_mean,
                        spec = spec
                      )
  )
}
```

The estimation function provides the method to calculate the mean from the full data. The bootstrap evaluation function provides the method for calculating the bootstrap representations; this is necessary due to the data structure in which rsample returns bootstrap samples.

Implementing a new model or statistic merely requires the developer to write these two functions and merge them into Turner's code base. This means that new features can be added very quickly and easily, providing myriad opportunity to extend Turner's usefulness. The source code for Turner is, furthermore, hosted online within a GitHub repository allowing for easy collaboration and updates as necessary.

3.4 User Experience Design

Turner was developed to make bootstrap procedures more accessible to lay people untrained in statistical methods, which necessitated special attention given to the UI, controls, and other elements of the user experience. The goal was to develop an app which could be understood and utilized easily by someone without other experience with the procedures while still providing useful insights into their model.

3.4.1 Diane from Marketing

During development, we considered a theoretical user we called "Diane from Marketing" - a white-collar professional with her own domain expertise, but without particular experience working with advanced statistical methods. Perhaps Diane took a basic statistics course in the past where she learned about such basics as means, standard errors, linear regression, etc., but she has not come across the bootstrap, jackknife, outlier detection, or other topics.

Despite this limited (but nonzero) extent of her knowledge, Diane has some statistic or model for which she needs to approximate a distribution; perhaps she is performing linear regression in the presence of outliers and must use a robust fit, or she needs a confidence interval for a median of some parameter of a market survey before moving forward with a new initiative. These very possible business requirements can be solved by using the bootstrap, but Diane will need an app like Turner to make such tools accessible.

3.4.2 Black Box Design

Since Turner was designed for users without detailed knowledge of the inner workings of the bootstrap we decided to encapsulate the "working parts" of the procedure and shield the user from them; essentially, we put our bootstrap implementation in a black box. As far as the end user is concerned, Turner is simply an app into which they put a data set into and out of which come results. The details of how such results are irrelevant to that end user, beyond knowing that bootstrapping was used on a given model.

This, of course, denies the user insight into exactly how the bootstrap achieved its goal, but the idea of Turner is to automatically run the diagnostic procedures internally so that the user does not *need* to have these insights. When the bootstrap results can be checked solely based on the app's provided outputs, the user's knowledge of exactly what occurred internally is irrelevant.

3.4.3 Workflow

The Turner workflow is very straightforward since Turner encapsulates the complicated parts of the procedure and shields the user. Before consulting Turner, users must collect data and decide on a model to fit or statistic to calculate; these tasks are external to Turner's purpose. Once the user has selected their model or statistic, they load their data into Turner as a comma-separated values (CSV) file using the online interface; example data sets are also included by default.

After data has been provided, the user specifies their selected model or statistic. The model or statistic is chosen through a drop-down menu, compactly displaying the available options to eliminate any need for the user to memorize the possibilities or refer to external documentation. Each option then has its own set of parameters, such as the variable for which to compute a mean or the fitting method for a linear regression; the parameters vary from model to model and statistic to statistic, so the controls for specifying them vary as well.

The user then indicates to the app that they are ready to proceed by clicking the appropriate button, and Turner goes to work. It displays an animated busy message as it carries out the bootstrap procedure and jackknife-after-bootstrap diagnostics, then shows the results once the computations are complete. There are two parts to the results display: the approximated distribution(s) from the bootstrap procedure itself and the jackknife-after-bootstrap results with outlier identification. Each model parameter estimator has its own results module with:

1. A histogram of the bootstrap replications of the parameter obtained during the procedure, with estimated density curve;
2. Point estimates of the estimator's standard error and bias;
3. Estimates of the distribution's skewness and kurtosis;
4. A bootstrap confidence interval, computed via the percentile method, with a slider control which allows the user to specify the desired value of α for calculating the confidence level of the $(1-\alpha)100\%$ interval.

These results displays give the user a good picture of the approximated distribution(s) obtained by bootstrapping in a clear, simple manner.

The jackknife-after-bootstrap results display page is shown after the bootstrap results. This page displays two elements: the jackknife-after-bootstrap plot (detailed in Section 2) and a list of cases which are observed to lie outside the jackknife-after-bootstrap uncertainty bands and are suspected to be outliers. The jackknife-after-bootstrap plot is interactive, meaning users can zoom and scroll for clarity as well as hover over points on the plot to view details of the case deleted at that point. The plot also includes a slider control for selecting the quantile for which to calculate the points on the plot. The list of outliers is simply displayed as a table where each row is a case and each column is a variable of that case.

3.5 Limitations and Future Developments

Naturally, Turner is not a do-it-all app; given the breadth and diversity of bootstrap methods and statistical models and the pace at which new ones are developed, attempting to develop such an app is a Sisyphean task. Given the extremely brief period of time allotted for app development we opted to design a very simple, polished prototype to showcase the feasibility of statistical software for bootstrapping and democratizing data science.

Possibly the greatest limitation of Turner is the small set of supported models and statistics. This, obviously, limits the utility of the app and the number of problems it can be used to solve. However, the modular design of the app makes it trivial to add support for a new model or statistic, provided that the developer is already able to build an R implementation of their model. Furthermore, the open-source nature of the R language lends itself to modular designs such as Turner's by encouraging the sharing of code such as Turner models; widespread adoption of an open-source data science tool such as Turner could give rise to communities sharing and collaborating to expand functionality even in the absence of the original developers.

Another limitation lies with the ways in which Turner automates the bootstrap process and its diagnostics. The bootstrap is a widely-used tool and, naturally, has many different versions working in many different ways. Turner currently supports only a very small subset, again due to development constraints. As with the supported models and statistics, the open-source platform and modular design is a saving grace as implementing and sharing new bootstrap methods is a trivial affair. Adding additional diagnostics, however, is less trivial. Future development plans for Turner include adding assessments of the impact of parameter nonpivotality on the calculations as well as corrections for internal error.

Further development on Turner need not be restricted to fixing shortcomings, however. Additional features can be added to increase Turner's utility in use such as APIs for interoperability with other statistical software, automatic report generation detailing the bootstrapping process and its results, parallel computing capability, and other enhancements. The exact nature of these future enhancements is, obviously, dependent on factors such as UX testing and end-user feedback.

4 Conclusion

Bootstrap methods provide a simple and widely-applicable way to assign measures of uncertainty such as standard error, bias, and confidence intervals to statistics for which they would otherwise be difficult or impossible to calculate. There are many variations of the bootstrap for varied situations in which it may be employed as well as different diagnostic procedures for assessing the results. However, the selection and employment of these various methods may be difficult for people without considerable training.

In this paper, we introduced a prototype of a novel web app called Turner which provides a simple interface for bootstrap methods and automates diagnostic analysis of the results. This app, when given a user specification for a model or statistic of interest, selects the appropriate bootstrap method through a combination of model foreknowledge and statistical testing before running the procedure within a "black box", automatically diagnosing problems, and displaying results to the user. This allows users without significant statistics training to perform bootstrapping to solve problems without consuming other data science resources in their organizations.

Turner demonstrates the utility of statistics and data science software developed for and targeted at a lay audience. While the skills needed to carry out such analyses are rare in the workplace, the problems requiring them are not; one solution to this problem is enhancing the capabilities of non-experts in statistical and data science methodology using software solutions. Naturally, these software solutions require integrated diagnostics and will carry some limitations, but it is possible to develop apps and programs to bring data science capabilities to the masses in their own fields of expertise.

References

[1] A. J. Canty, A. C. Davison, D. V. Hinkley, and V. Ventura. Bootstrap diagnostics and remedies. *Canadian Journal of Statistics*, 34(2):5–27, 2006.

[2] Winston Chang, Joe Cheng, JJ Allaire, Yihui Xie, and Jonathan McPherson. *shiny: Web Application Framework for R*, 2020. R package version 1.5.0.

[3] J. Cornelissen. The democratization of data science. *Harvard Business Review*, 2018.

[4] B. Efron. Bootstrap methods: Another look at the jackknife. *The Annals of Statistics*, 7(1):1–26, 1979.

[5] B. Efron. Jackknife-after-bootstrap standard errors and influence functions. *Journal of the Royal Statistical Society: Series B (Methodological)*, 54(1):83–111, 1992.

[6] B. Efron and R. Tibshirani. Bootstrap methods for standard errors, confidence intervals, and other measures of statistical accuracy. *Statistical Science*, 1(1):54–77, 1986.

[7] B. Efron and R. Tibshirani. *An Introduction to the Bootstrap*. Chapman and Hall, 1993.

[8] R. A. Fisher. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7(2):179–188, 1936.

[9] T. Hastie, J. Friedman, and R. Tibshirani. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2 edition, 2017.

[10] H. V. Henderson and P. F. Velleman. Building multiple regression models interactively. *Biometrics*, 37(2):391–411, 1981.

[11] Max Kuhn and Hadley Wickham. *Tidymodels: a collection of packages for modeling and machine learning using tidyverse principles.*, 2020.

[12] M. H. Quenouille. Notes on bias in estimation. *Biometrika*, 43(3/4):353–360, 1956.

[13] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2021.

[14] Julia Silge, Fanny Chow, Max Kuhn, and Hadley Wickham. *rsample: General Resampling Infrastructure*, 2021. R package version 0.0.9.

[15] H. White. A heteroskedasticity-consistent covariance matrix estimator and a direct test for heteroskedasticity. *Econometrica*, 48(4):817–838, 1980.