The Dagstuhl Beginners Guide to Reproducibility for Experimental Networking Research

Vaibhav Bajpai
TU Munich

Anna Brunstrom
Karlstad University

Anja Feldmann
Max Planck Institute for Informatics

Wolfgang Kellerer
TU Munich

Aiko Pras
University of Twente

Henning Schulzrinne
Columbia University

Georgios Smaragdakis
TU Berlin

Matthias Wählisch
FU Berlin

Klaus Wehrle
RWTH Aachen University

ABSTRACT

Reproducibility is one of the key characteristics of good science, but hard to achieve for experimental disciplines like Internet measurements and network systems. This guide provides advice to researchers, particularly those new to the field, on designing experiments so that their work is more likely to be reproducible and to serve as a foundation for follow-on work by others.

CCS CONCEPTS

• **General and reference → Surveys and overviews**;

KEYWORDS

Experimental networking research; Internet measurements; Reproducibility; Guidance

1 INTRODUCTION

Good scientific practice makes it easy for researchers other than the authors to reproduce, evaluate and build on the work. Achieving these goals, however, is often challenging and requires planning and care. We attempt to provide guidelines for researchers early in their career and students working in the field of experimental networking research, and as a reminder for others. We begin by summarizing the terminology (§ 1.1) that will be used throughout this article. We then elaborate the goals and principles (§ 1.2), describe best practices required for reproducibility in general (§ 2) and for specific research methodologies (§ 3), provide tool recommendations (§ 4) and point to additional resources (§ 5).

1.1 ACM Terminology

The terms repeatability, replicability and reproducibility are often used interchangeably and may not necessarily be used consistently within or across technical communities. Since the Association for Computing Machinery (ACM) [1] publishes a significant fraction of papers in networked systems and Internet measurements, we draw on their definitions and summarize them in Table 1.

Table 1: Repeatability, replicability, and reproducibility as defined by ACM [1].

	Level of change	
Term	Team	Setup
Repeatability	same	same
Replicability	different	same
Reproducibility	different	different

Repeatability is achieved when a researcher can obtain the same results for her own experiment under exactly the same conditions, i.e., she can reliably repeat her own experiment ("Same team, same experimental setup").

Replicability allows a different researcher to obtain the same results for an experiment under exactly the same conditions and using exactly the same artifacts, i.e., another independent researcher can reliably repeat an experiment of someone other than herself ("Different team, same experimental setup").

Reproducibility enables researcher other than the authors to obtain the same results for an experiment under different conditions and using her self-developed artifacts ("Different team, different experimental setup").

1.2 Goals and Principles

One of the fundamental hallmarks of science is that research results produced by one team can be replicated or reproduced

(§ 1.1) by another team. Ideally, the second team should only need their general knowledge of the discipline and the details provided in the published paper, complemented by auxiliary materials such as software documentation or technical reports in some cases.

However, repeatability, replicability, and reproducibility is about more than just following the scientific method and being a "good research citizen". By carefully documenting your work flow and following best practices, other team members in your research group can continue earlier work and build on it. Often, you yourself will need to revisit earlier work, e.g., when compiling your research for your dissertation or a journal paper, recreating results or updating them to reflect new related work or changes in the environment. Nobody likes spending time on reverse-engineering your own code written a year ago or code written by somebody else, figuring out why software packages do not compile or wondering whether you can trust the experimental data you gather. Besides facilitating progress in science, following best practices will also make mistakes less likely or at least easier to find.

The practices described below work best if followed early on, not just as the final step when completing a project.

2 GENERAL BEST PRACTICES

Long before you write a paper, the following best practices help to ensure that your research succeeds and that you can trust your results.

2.1 Problem Formulation and Design

Hypothesize: "Think first, run later": Formulate and document your hypothesis, design the experiments to validate (or not) the hypothesis, conduct the necessary experiments, and finally check the hypothesis. Indeed, often the outcome of an experiment should lead you to revisit the hypothesis. But sometimes, if an experiment does not give you the predicted results or gives you results that seem a little too good to be true, this may be due to a mistake in the analysis chain. Therefore, each step needs to be validated and cross-checked. As such it is good practice to double check results with others who may be able to spot problems, e.g., your advisor, someone from the organization responsible for the infrastructure on which the data was gathered, or the author of a software component you used. If you work in a small team, it is a good idea to plan the work so that different persons work on different results so that each one can cross check the work of the others.

Plan and solicit early feedback: Plan and prototype how you want to present your results as early as possible. Visualizations are necessary to explain your results, but they also help you spot anomalies. You should be able to explain notches, spikes or gaps in your graph by something beyond randomness. Follow guidelines for exploring the parameter space, e.g., an ANOVA [18] experimental design. Get feedback early and often: before you start your project, after your initial experimental design, after your first small-scale results and after your first large-scale results.

Iterate: You will likely end up having to redo steps as you modify the system under test or improve your measurements and data analysis scripts. Record steps and automate them, e.g., in scripts or Makefiles, so that you are less likely to forget to set a command line parameter, for example. How often do you need to repeat your measurements to eliminate transient factors and gain confidence? Especially when measuring operational systems such as data centers or the Internet, one-time measurements are prone to be biased by transient effects, temporary congestion or just the particular time of day. Those factors should be accounted for when actually planning the measurement.

Factor dynamism: Generally expect that operational systems you are measuring against are not static during your measurements. There is evidence that well-known Internet services change constantly and that there are ongoing experiments run by service providers that may interfere with your own measurements.

2.2 Documentation

Record the experiment: Documenting *all* steps and observations is critical. Scientists in the natural sciences keep lab notebooks for a reason — follow their example. The lab notebook can be an electronic shared document, recording each step and each resulting observation. Record mistakes, too, so that others do not have to repeat them. If the lab notebook is electronic, recording script executions can be a first step to automating the workflow. It is often tempting to skip documenting code until later, when there is supposedly more time, but that time never seems to occur. Research artifacts often live longer than you anticipate and may be shared with other members of the research team. Thus, code as if you are your colleague who has to pick up your project.

Treat metadata as data: Any data file or database needs to be accompanied by meta data to help you and others understand how the data was created, what it contains, where to find its documentation, and how to recreate it. Meta data can be conveyed via file naming, contained in header sections in the data, or stored separately in a data log that references file names and, to avoid accidental file name reuse, file hashes. Consider automating the generation of the "mechanical" meta data in the scripts or tools you write, preferably in some machine-readable format such as JSON or XML.

Use a version-control system: Using a version control system for code, documentation, paper text, as well as experimental results is essential. This will help you determine if a change in measured results might be due to an innocent-looking code change and which experiments you might need to run again. Whenever possible, you create a release of your own software that you used to create the publishable results. Note that including the raw experimental data may or may not be feasible due to size, privacy, or other constraints.

Keep regular backups: Keep backups. There is nothing more upsetting than losing the original data of a paper that you are about to publish or that already got published. This also avoids digging into the file systems of graduate students who have long left the university and hoping that their account has not been deleted. Indeed, the data management plans for most organizations and research grants require that scientific artifacts are not only documented but also preserved for multiple years (e.g., five to ten years). Most research institutions offer resources to store data safely and with flexible access control policies.

2.3 Experimentation and Data Collection

Validate and scale: Start small and then expand. Run small sample sets, where you can readily predict the results, to understand and verify your tools, approaches and analysis setup. These can then later be used as test cases and sanity checks to ensure that the analysis pipeline is still working even if one of the components gets updated. Use a tool chain to first validate previously published results to ensure that there are no fundamental flaws in the analysis or your understanding of the problem. A welcome side effect is that this often leads to insights which lead to new research results.

Do not reinvent the wheel: Before initiating a major software development project check if there is a tool that solves your problem. Creating your own tool may bring you to face issues that others have already solved. More than that, creating your own tool also likely commits you to maintaining it. Think about convenient ways of decomposing your problem to follow the Unix philosophy of building simple, modular, and extensible code that can be easily maintained, tested and re-purposed.

Monitor your experiment: Make sure to monitor your toolchain, preferably by automated checking tools. Common problems include running out of disk space and, therefore, creating zero-length files; reboot of a machine without restarting the tools or causing log files to be overwritten; wrong permissions, e.g., when access tokens time out; network failures and, therefore, missing results from a remote machine or API and finally, resource leaks, such as too many open files, that prevent or distort data gathering.

2.4 Handling Data

Data privacy and anonymization: Most datasets have privacy constraints that you need to respect. You should never try to de-anonymize data, as that is unethical and will likely discourage others from making data available. Before making data available to others, consider whether it raises any privacy concerns and whether these concerns can be alleviated by anonymization. If in doubt, always consult other members of your research team, more senior researchers and your local ethics panel or institutional review board (IRB). Data that may seem unlinkable by itself can now often be de-anonymized by drawing on external data sources.

Data integrity: Check for the integrity of your data and account for observation biases. Did you consider synchronization between system elements, randomization, the effects of caching? When evaluating the performance of a system, will likely use cases depend on the average, best or worst-case performance or some "likely" worst-case performance?

Licensing and giving credit: Consider early how code you use or write will be licensed. Can you share copyrighted code that you purchased or have access to through your institution with your team or the public? Does everyone on your team agree with how you intend to license code you wrote? (For instance, your role in the institution may determine whether your code is for-hire work or your own.) Does the code license require you to make modifications publicly available? Do code or data use Creative Commons [10] licenses that mandate giving credit to sources? Does your research institution or the organization providing research support have guidelines you need to be aware of? For example, some research funding agencies strongly encourage giving credit to their funding, using template text. Consider that often the most restrictive software license for a system determines whether others can use it. But even restrictive code licenses do not prevent sharing of output data or results.

3 WHAT SHOULD BE DOCUMENTED?

Each paper or thesis should document key experimental conditions, possibly in an appendix or separate technical report for lengthy descriptions of details. Many of these experimental conditions that are needed to make your work reproducible are similar for all basic types of experimental networking research, often used in combination: simulation (§ 3.1), prototyping (§ 3.2), network measurements (§ 3.3) and human factors experiments (§ 3.4). We describe considerations for each methodology in turn below.

3.1 Simulations

Simulation is a well-known method to understand and validate a proposed concept, protocol or a system. When simulating a system under test (SuT), a model of this SuT is used and its behavior under varying input and configurations analyzed. Your analysis depends completely on the chosen model and will only reflect the characteristics of the model. Therefore, choose your model with care – whether you create it yourself or use the model somebody else created. Furthermore, consider the granularity at which you plan to simulate, such as traffic flows, individual packets or the physical channel model.

In order for someone to *repeat* your simulation results, your simulation code and input data should be well packaged and documented such that someone can easily re-run your simulation, e.g., by just executing a Makefile or script. In order to be able to *reproduce or replicate* your results, other researchers should also understand why you chose the particular simulation parameters.

Software setup: Describe the simulation software, including the version and required run-time environment. Which additional tools such as traffic generators, topology models, analysis tools are required? Which versions were used? Does your simulation require any specific run-time or execution environment, such as many cores or massive amounts of RAM, that may exceed what is commonly available?

Data input and configuration: Describe the network or system topology including transmission rate, bit error rates and propagation delays. What traffic traces or models did you use? What were the parameters of the models, including units? (Be particularly careful with easily confused units, such as kb/s (kilobits [1000 bits] per second) vs. KB/s (Kilobytes (1024 bytes) per second).) If you are including a model of the physical channel, such as a wireless link, what parameters did you choose and are they meant to represent a particular real-world environment? If aspects of your traffic or system parameters are chosen randomly, describe which and how you generated the random variables. If random number generator seeds matter, provide them. Any simulator configuration file that can be shared?

Limitations: Is your simulation limited in some important way, e.g., in terms of scale or the execution time needed? How does your simulation abstract and simplify the system you are modeling?

Experiments: How often did you repeat the experiment and how did you choose the repeat count? How did you initialize the system, e.g., were caches cleared before each run? How did you space your parameters? Did they cover the desired design space for your system?

Analysis: In general, data is sacrosanct and all raw data should be archived. How did you prepare the data? Did you remove any outliers or obvious measurement errors? Did outages or errors leave gaps in your data gathering? How are you accounting for start-up and transient effects? Were there any anomalies? How are you showing the strength of your evidence, e.g., by confidence intervals, variance, ANOVA, goodness-of-fit testing? How did you choose the parameters for statistical tests? Did you change your measurement approach to, for example, meet a p-value or confidence interval threshold? If you are testing a hypothesis, how strong is the evidence that the results are not due to random chance?

Presentation: Did you include all units for all axes in a clear and unambiguous way? Captions for plots should explain the setting and contain all major parameters so that the caption and figure can stand alone. Consider data formats that allow including the plot points or complement plots with tables showing raw data in an appendix or an extended technical report.

Data access: If your simulation depends on input data other than parameterized random variables, such as traces or topologies, these should be included with the simulation code or stored in a publicly accessible repository – see § 3.2.

3.2 Systems Prototyping and Evaluations

To evaluate a new protocol, service or algorithm you can build a prototype and then measure its scalability, performance or efficiency, typically in a controlled environment such as a testbed.

Software setup: Describe the operating system, any non-standard libraries, including version information, and the hardware environment, including network interfaces, memory size and graphics cards. For libraries, note if these are not readily available, e.g., due to licensing restrictions. If you used an emulator (e.g., for network links), describe the configuration in detail.

Data input and setup: What data sources drove the input for your system? What were sources of randomness?

Limitations: Are you aware of any limitations in your system that may have influenced the measurements, such as performance limitations of the hardware, other experiments sharing the same infrastructure, caches or timing resolution and clock synchronization between systems?

Experiments: How often did you repeat the experiment? What was the set of parameters you used? (As above, be careful to use unambiguous units and explain if necessary.)

Analysis and presentation: See § 3.1.

Data access: Are any of the traces or raw data available to others? Did you document the log or trace file format? Is it unambiguous which data trace or log correspond to which experiment or measurement? Is the data public or restricted, for instance under non-disclosure agreement (NDA)? Do you anticipate that the data will only be available for a limited time, e.g., because it is a rolling data collection? Consider getting a Digital Object Identifier (DOI) for your data set to make it easy to reference.

3.3 Real-world Measurements

Measurements help understand how real systems function. For example, research might measure the current state of deployment of a protocol or feature in the Internet, the characteristics of Internet usage or the behavior of congestion control, security and routing protocols. Measurements can also complement simulations by observing how well a proposed system or protocol functions in the Internet or a real campus or data center network. Measurements can be intra- and inter-domain, measuring the whole Internet, one or more Internet service providers, or a single data-center. Unlike for the previous case, you typically have very limited control over your measurement environment.

Setup: Where were your measurement vantage points? For Internet measurement points, what kind of networks were they located in? Do you know the service provider, organization, access technology or geographic location? How did you choose them? For many measurements, the number and location of the measurement vantage (observation) points determines whether the results you obtain are only narrowly or more broadly applicable.

What software did you use to collect the data, e.g., IPFIX [9], Netflow, `traceroute`, your own mobile application? Did you rely on a public measurement infrastructure, e.g., RIPE Atlas [16]; Planetlab [8]; etc. Describe the software version and execution environment, such as the operating system and any relevant libraries. What hardware (vendor, model, version or model year) did you use, including any special network interfaces, dedicated flow exporters or special-purpose switches? Do your measurements rely on precise time and how did you ensure clock synchronization both between measurement points and to absolute time?

When running active measurements, characterize your traffic sources. For passive measurements, describe whether you collected all traffic or sampled traffic.

Data collection: Do the measurements represent a snapshot in time or a longitudinal observation? Justify your sampling period (e.g., subset of packets versus comprehensive packet capture), the frequency of data collection (e.g., hourly, daily, randomly), and the number of times the data collection has been repeated.

Time and date may influence your results. When was the measurement collected? Be sure to clearly state the timezone. While UTC is generally preferred, in cases where your measurements depend on human diurnal cycles, it may be helpful to capture the local time.

Document all external data sources, such as routing tables, that you collected or that are provided by third parties. If the additional data sources do not describe the same time interval or locations as your collected data, mention this and justify why you consider the data to be applicable. Furthermore, when you measure in an open system, such as the Internet, which is subject to uncontrolled changes, you need to collect all relevant data about the system itself during the measurements. This requires much more planning of the measurements compared to a controlled lab testbed setup where the system aspects are mostly static and can likely be inspected after the measurements have finished. For example, if you work with the Alexa 1M most-popular web site lists, it should be clear which version of the list you actually used. But even then there is a dynamic mapping of names to addresses using the DNS — it may matter where, when and how you resolve the names to addresses. If you use a distributed set of vantage points, you will sooner or later need to understand the topology as seen from the perspective of the vantage points. Hence, it is best to collect `traceroute` data (and if relevant name resolution data) with your measurements as this will be crucial later on to interpret your data set.

Any missing data needs to be mentioned, particularly data gaps in the collection of measurements caused by operational outages or system maintenance.

Limitations: Are there limitations that may affect the validity or accuracy of your measurement data or may bias your results?

Analysis and presentation: See § 3.1.

Data access: See § 3.2.

Ethics considerations: Do your measurements implicate potential ethical concerns, in particular those that anybody reproducing your work may need to be aware of? For example, you should document any constraints imposed by institutional review boards or ethics committees. This will also help reviewers judge whether you are complying with general community guidelines [3, 11], or those of conferences such as ACM *Internet Measurement Conference* (IMC).

3.4 Human Subject and Subjective Experiments

In subjective experiments, participants evaluate the usability or quality of experience (QoE) of a service, functionality, or software. Often, you are testing a hypothesis ("my system works better than the old system", "Variable X improves task performance"), which should be formulated ahead of time.

Setup: Who were the experimental subjects, e.g., by age brackets, gender, education and computing skills? Had the subjects taken part in similar experiments before? How did you solicit volunteers? If applicable, note the tracking number for your IRB (Institutional Review Board) or ethics committee approval.

Experiments: Describe how the experiment was conducted. Were the subjects provided with instructions or just handed your artifact? Were they asked to complete specific tasks? Did the subjects communicate with each other or perform tasks independently?

Limitations: How did your experiment deviate from "real life", e.g., in duration or nature of the task?

Analysis and presentation: See § 3.1.

Ethics considerations: Human subject experiments will likely require approval by an institutional review board (IRB) or ethics panel. You should document key considerations for protecting human subjects that anybody replicating your study should be aware of and make your IRB filing available to others. (Following the same process during a replication does not relieve the replicator from the duty of seeking approval from an IRB or ethics panel, nor does it guarantee that such approval will be granted.)

4 TOOL RECOMMENDATIONS

Try to use common tools that are widely and readily available. Only develop your own tools if there are no reasonable alternatives. Writing good tools almost always takes longer than you were initially planning for and you will have to maintain those tools for both yourself, other team members and other researchers trying to reproduce your results. We have found the following tools useful in experimental networking research: Document your work in shared lab notebooks using Jupyter. Package your software as containers, using Docker and Kubernetes, or virtual machines (using Vagrant) to avoid dependency hell and ease execution in different environments. Use version control (with Git and its ecosystem such as Github and Gitlab) for software and scripts, including scripts for plotting (e.g., based on Python `matplotlib`, `R` or `gnuplot`). You may also want to create software releases by using Git tags.

Many research institutions offer centralized resources for storing experimental data for the long term. Do not store any valuable experimental data on personal devices, laptops or computing devices used for experiments. An experiment may accidentally trash a disk or brick the machine. Your lab will eventually retire old computers and you do not want to have to guess which of the terabytes of data are still valuable.

5 ADDITIONAL RESOURCES

The proceedings of the ACM SIGCOMM 2003 Workshop on Models, Methods and Tools for Reproducible Network [6] and the ACM SIGCOMM 2017 Workshop on Reproducibility [5] summarize past discussions on this topic. The Stanford University Reproducibility course [19] is a good example of how students can take published research and attempt to reproduce and document the findings. A list of accepted papers in SIGCOMM-sponsored conferences that released artifacts were recently (2017) surveyed and a compiled list has been made available [12]. The recently established SIGCOMM Artifacts Evaluation Committee carried this initiative forward and applied badges to accepted papers in SIGCOMM-sponsored events in 2018. For instance, CoNEXT 2018 has published the badges of these accepted papers both in the conference proceedings and on the conference web page. Such lists of papers with released artifacts and badged papers can be a good starting to point for students to get started with reproducing published research. Papers by Allman [2] and Reuter et al. [13] discuss the dynamism and heterogeneous nature of the Internet. Other interesting papers highlight pitfalls with IP-address-based geolocation [17], the popularity of webpages [15], and `traceroute` [4] that our community has learned from over the past years of empirical research. Other scientific communities are engaging in extensive efforts to improve replicability and reproducibility [7, 14].

Acknowledgments

The ideas in this paper were developed at the Dagstuhl Seminar #18412 on "Encouraging Reproducibility in Scientific

Research of the Internet" that took place in October 2018. Jürgen Schönwälder and Olivier Bonaventure provided valuable input to this manuscript.

REFERENCES

[1] ACM. 2018. Artifact Review and Badging. https://www.acm.org/publications/policies/artifact-review-badging. (2018). Reviewed April 2018; accessed November 11, 2018.

[2] Mark Allman. 2013. On Changing the Culture of Empirical Internet Assessment. *ACM SIGCOMM Computer Communication Review (CCR)* 43, 3 (July 2013), 78–83. https://doi.org/10.1145/2500098.2500110.

[3] Mark Allman and Vern Paxson. 2007. Issues and Etiquette Concerning Use of Shared Measurement Data. In *Proceedings of the 7th ACM SIGCOMM Conference on Internet Measurement*. ACM, San Diego, California, USA, 135–140. https://doi.org/10.1145/1298306.1298327

[4] Brice Augustin, Xavier Cuvellier, Benjamin Orgogozo, Fabien Viger, Timur Friedman, Matthieu Latapy, Clémence Magnien, and Renata Teixeira. 2006. Avoiding traceroute Anomalies with Paris traceroute. In *Proceedings of the 6th ACM SIGCOMM Conference on Internet Measurement*. ACM, Rio de Janeriro, Brazil, 153–158. https://doi.org/10.1145/1177080.1177100

[5] Olivier Bonaventure, Luigi Iannone, and Damien Saucez (Eds.). 2017. *Reproducibility '17: Proceedings of the Reproducibility Workshop*. ACM. https://doi.org/10.1145/3097766.

[6] Georg Carle, Hartmut Ritter, and Klaus Wehrle (Eds.). 2003. *MoMeTools '03: Proceedings of the ACM SIGCOMM Workshop on Models, Methods and Tools for Reproducible Network Research*. ACM, New York, NY, USA. https://doi.org/10.1145/944773.

[7] Center for Open Science. 2018. Open Science Framework: A scholarly commons to connect the entire research cycle. https://osf.io/. (2018). accessed November 11, 2018.

[8] Brent Chun, David Culler, Timothy Roscoe, Andy Bavier, Larry Peterson, Mike Wawrzoniak, and Mic Bowman. 2003. PlanetLab: An Overlay Testbed for Broad-coverage Services. *ACM SIGCOMM Computer Communication Review (CCR)* 33, 3 (July 2003), 3–12. https://doi.org/10.1145/956993.956995

[9] Benoit Claise, Brian Trammell, and Paul Aitken. 2013. *Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information*. RFC RFC 7011. RFC Editor, Fremont, CA, USA.

[10] Creative Commons Corporation. 2018. Creative Commons. https://creativecommons.org/. (2018). accessed December 31, 2018.

[11] David Dittrich and Erin Kenneally. 2012. *The Menlo Report: Ethical Principles Guiding Information and Communication Technology Research*. Technical Report. Department of Homeland Security. https://www.caida.org/publications/papers/2012/menlo_report_actual_formatted/

[12] Matthias Flittner, Mohamed Naoufal Mahfoudi, Damien Saucez, Matthias Wählisch, Luigi Iannone, Vaibhav Bajpai, and Alex Afanasyev. 2018. A Survey on Artifacts from CoNEXT, ICN, IMC, and SIGCOMM Conferences in 2017. *ACM Computer Communication Review (CCR)* 48, 1 (2018), 75–80. https://doi.org/10.1145/3211852.3211864

[13] Andreas Reuter, Randy Bush, Italo Cunha, Ethan Katz-Bassett, Thomas C. Schmidt, and Matthias Wählisch. 2018. Towards a Rigorous Methodology for Measuring Adoption of RPKI Route Validation and Filtering. *ACM SIGCOMM Computer Communication Review* 48, 1 (2018), 19–27. https://doi.org/10.1145/3211852.3211856

[14] rOpenSci. 2018. Transforming science through open data and software. https://ropensci.org/. (2018). accessed November 11, 2018.

[15] Quirin Scheitle, Oliver Hohlfeld, Julien Gamba, Jonas Jelten, Torsten Zimmermann, Stephen D. Strowes, and Narseo Vallina-Rodriguez. 2018. A Long Way to the Top: Significance, Structure, and Stability of Internet Top Lists. In *Proceedings of the Internet Measurement Conference 2018*. ACM, Boston, Massachusetts, USA, 478–493. https://doi.org/10.1145/3278532.3278574

[16] Vaibhav Bajpai and Jürgen Schönwälder. 2015. A Survey on Internet Performance Measurement Platforms and Related Standardization Efforts. *IEEE Communications Surveys and Tutorials* 17, 3 (2015), 1313–1341. https://doi.org/10.1109/COMST.2015.2418435

[17] Zachary Weinberg, Shinyoung Cho, Nicolas Christin, Vyas Sekar, and Phillipa Gill. 2018. How to Catch when Proxies Lie: Verifying the Physical Locations of Network Proxies with Active Geolocation. In *Proceedings of the Internet Measurement Conference 2018*. ACM, Boston, Massachusetts, USA, 203–217. http://doi.acm.org/10.1145/3278532.3278551

[18] Wikipedia contributors. 2018. ANOVA: Analysis of Variance. https://en.wikipedia.org/wiki/Analysis_of_variance. (2018). accessed November 30, 2018.

[19] Lisa Yan and Nick McKeown. 2017. Learning Networking by Reproducing Research Results. *ACM SIGCOMM Computer Communication Review (CCR)* 47, 2 (2017), 19–26. https://doi.org/10.1145/3089262.3089266