CHAPTER

11

Analyze Data and Observations

Finally. You have completed the testing and are now ready to dive in and transform a wealth of data into recommendations for improvement. Typically, the analysis of data falls into two distinct processes with two different deliverables:

- The first is a preliminary analysis and is intended to quickly ascertain the *hot spots* (i.e., worst problems), so that the designers can work on these immediately without having to wait for the final test report. This preliminary analysis takes place as soon as feasible after testing is complete. Its deliverable is either a small written report (which you could deliver by email, as a short slide deck, on a blog, or to a groupware site) or a verbal presentation of findings and recommendations. The steps for performing this preliminary analysis are covered in this chapter:
  - Compile data.
  - Summarize data.
  - Analyze data.

  Generally, the point is to get out the weeds to be able to see the larger trends and patterns.
- The second is a comprehensive analysis, which takes place during a 2- to 4-week period after the test. Its deliverable is a final, more exhaustive report. This final report should include all the findings in the preliminary report, updated if necessary, plus all the other analyses and findings that were not previously covered. The steps for generating the final report are covered in Chapter 12.

A word of caution is in order regarding preliminary findings and recommendations. Developing and reporting preliminary recommendations creates a predicament for the test moderator. Your recommendations must be timely so that members of the development team, such as designers and writers, can begin implementing changes. However, you also need to be thorough, in the sense of not missing anything important. Once preliminary recommendations are circulated for public consumption, they quickly lose their *preliminary* flavor. Designers will begin to implement changes, and it is difficult to revisit changes at a later time and say ''Oops, I don't really think we should change that module after all.''

You could simply avoid producing preliminary recommendations, but if designers viewed the tests, they are sure to act on what they saw prior to your final report. Therefore, not providing *any* preliminary recommendations is not a satisfactory option. The best compromise is to provide preliminary findings and recommendations but be cautious and err on the conservative side by providing too little rather than too much. Stick to very obvious problems, such as ones that prevented completion of the task by all or the majority of the users. If you are unsure about a finding or a recommended solution without performing further analysis, simply say so.

As outside consultants, we typically meet with the development team immediately after the test for a debriefing and informal report, and then follow up with a formal report later. The informal report will only include obvious items that do not require further analyses, and if possible, come out of consensus debriefings that we held throughout the study or after the last session. In addition, we always qualify our preliminary recommendations as ''*preliminary*,'' and if we leave any paperwork behind, it is clearly marked **PRELIMINARY** in large letters. This gives us the option of changing a recommendation later based upon further analysis.

Now let's move on and discuss the steps involved in analyzing test data and developing recommendations. There are four major steps to the process.

1. Compile and summarize data.
2. Analyze data.
3. Develop recommendations.
4. Produce the final report.

This chapter covers compiling and analyzing data. The next chapter describes how to turn the results into insights and findings for reporting.

Compile Data

Let's look at the process of compiling and summarizing the data you have collected. The following is a step-by-step process for compiling data, although in all honesty it never goes as smoothly and as linearly as described here. It tends to be circular and iterative, as some answers lead to more questions.

Begin Compiling Data as You Test

The process of compiling involves placing all the data collected into a form that allows you to see patterns. The compilation of data, whether you are creating a preliminary report or not, should go on throughout the test sessions. Not only does this speed up the overall analysis process, but it also serves as a checkpoint to see that you are collecting the correct data and that the data matches the problem statements in the test plan. Compiling data as you test also helps you to see if you are missing anything important, and that you understand what you have collected during one day before moving on to the next day.

At the end of each day, gather all the data for that day's session. Make sure that all data is legible, especially if others are helping you. If observers are assisting with collection, make sure that you can read their notes, too, and that they are collecting data as you expect. Begin to get recordings and data files backed up and audio recordings transcribed. It is much easier to work with digital recordings than analog ones. Having transcriptions can be helpful for protocol analysis or studies where you are particularly concerned with vocabulary or information architecture.

Transfer handwritten notes to a computer, and transfer times and other quantitative data onto a master sheet or computer spreadsheet. If you like, keep a running summary of the data, which you update each night. For example, tally errors and successes by task on a spreadsheet and recompute the average after each day's data is added.

This ongoing compilation of data takes advantage of the fact that the test is still fresh in your mind so that you can remember odd quirks and events that happened during the day. Even with the best note taking, keystroke or mouse click recording, and so on some events simply are not captured, except in your memory, and they will fade unless you record them. That is why we recommend doing some compiling each day of a study. The ongoing compilation also prepares you to move faster on any preliminary analysis that you provide.

If you happen to be conducting an iterative, fast-turnaround test (e.g., changes are made to the product after every few sessions), compiling data each night will be a necessity. You and the design team will have to decide which changes to the product to implement for the next day's session, and there should be a record of what those discussions were. In this situation, you may want to update your test plan, session script, and debriefing guide (see Chapter 8 for more about each of these) to reflect the revisions in the product.

Organize Raw Data

When you complete the sessions from a study, you probably have several types of data at hand, from recordings to notes, to questions and comments from participants, to issues lists from observers. The toolbox for organizing the raw data has many offerings. There are, of course, transcriptions of sessions, especially if you have conducted a ''think aloud'' study (see Chapter 9 for more information). There are various types of recordings to review (even the audio recordings can offer hints at emotion that can help you determine a severity rating for a problem). In addition, you may have used monitoring software that collected other data such as mouse clicks, and that should generate tables and spreadsheets for you.

At some point, though, you simply have to start going through your own notes or the notes you get from your assigned data gatherer and/or observers. The tools are simple but many: lists, tallies, matrices, stories, storyboards, structure models, flow diagrams, and so on. Use any tool that helps you get a handle on how the data looks. Break out highlighters, yellow sticky notes, and flip charts if you do better with moving things around physically rather than virtually.

One of our favorite tools is also one of the simplest and most available: a spreadsheet program. Whether you are defining and quantifying everything to be able to get statistical significance or you just want to know what the dominant task path is, you can drop text or tallies into the cells of a spreadsheet and then use its tools to help you with analyzing the data. A spreadsheet program helps you generate counts, averages, medians, means, and percentages. Obviously, it's good for helping with number-related analyses. However, it will also help you sort and filter data, which can lead you to recognizing important patterns that you may not be able to represent statistically.

Even laying text out in a spreadsheet — such as answers to interview questions — can help you see true duplication as well as near matches that reveal useful insights. For example, in one study Dana asked participants ''When you tried to use the site before but couldn't find what you were looking for, what did you do then?'' In reviewing a dozen responses, she could see that the ultimate answer wasn't ''go to the site map,'' as the team had hoped it would be, but instead was ''ask someone else'' because the participants said things like, ''I ask my manager,'' ''contact the help desk,'' or ''email a coworker.'' To analyze the text, Dana skimmed through the first few transcribed responses in a spreadsheet, looking for key words. Finding two or three, she used the Find and Replace feature of the spreadsheet to search for each keyword in turn, and changed that cell to a different color. The software automatically told her how many instances it had changed, and she could see which participants said what. Over the next few Find-Replace steps, Dana could track trends to user groups. Pretty simple, but efficient and effective.

See the web site that accompanies the book (`www.wiley.com/go/usabilitytesting`) for full-color examples of analysis spreadsheets and tallies.

Of course, you could also use outlining programs, data visualizers, or mind-mapping software. Use whatever works for you to be able to recognize trends and patterns. As those trends and patterns are revealed, you should be able to draw inferences about what the usability problems were.

Summarize Data

After all sessions have been completed, finish compiling the data. Transfer information from data collection sheets onto summary sheets. Or, if you are using an automated data logger that summarizes data for you, print out those files and review them. *By aggregating the data measures you have collected, you get a snapshot of what happened during the test — where participants performed well and where they performed poorly*. These summaries are also used to indicate if there were differences in performance of different groups, such as novice and experienced users, or differences in performance of different versions of a product. It is at this point that you begin to have what you need to determine whether the test met its objectives and to answer the original research questions outlined in the test plan.

Summarize Performance Data

You will want to summarize performance data in terms of errors and task accuracy. You may want to use timings, too. The most common *descriptive* statistics for these types of performance data are shown in the sections below. (We talk about inferential statistics later in this chapter.) Descriptive statistics are simply techniques for classifying the characteristics of your data that help you see patterns that may review problems or insights. All of the statistics use simple formulas that are available on most computer spreadsheet programs.

Task Accuracy

For task accuracy, you have several different statistics at your disposal. You could simply count the number of errors made per task. Or, you could go further and categorize the errors by type, such as errors of omission (leaving something out), errors of commission (doing something not needed), and so on. Or, as described in the following discussion, you could track the number of participants who performed successfully, either within the time benchmark expected or outside of it. You can also track those participants performing successfully but requiring some assistance. This statistic, however, is not all that helpful unless you have kept the type of assistance consistent from session to session and know where in the task and product the assistance was given.

Following are three types of statistics that relate to task accuracy:

- **Percentage of participants performing successfully, including those who required assistance.** This task accuracy measure includes every participant who completed the task successfully, regardless of whether the task was completed within a time limit, or if participants needed assistance. If this number is very low, say 50 percent, it indicates *very* serious problems with the product because participants could not perform successfully even with assistance and extra time.
- **Percentage of participants performing successfully.** This task accuracy statistic indicates the percentage of participants who were at least able to muddle through the task well enough to complete it successfully. If the participants made errors, you know they were eventually able to correct themselves and perform successfully.
- **Percentage of participants performing successfully within a time benchmark.** This task accuracy statistic is an indication of correct performance and efficiency. If 7 of 10 participants achieved success within the allotted time, then the task accuracy would be 70 percent. You and your team should have determined ahead of time what the benchmark should be (see Chapter 8). If you are conducting multiple usability tests over time as the design is iterated for elimination of usability problems, the team may allow that 70 percent success rate is acceptable in the first couple of rounds of usability testing. (For a discussion about whether 70 percent success is good enough, see Chapter 3.)

Figure 11-1 shows an example of a combined summary of task timings and task accuracy scores from a hypothetical test.

Note that one can look across any line on the table and see the timings and score for each task. For this particular summary, a score was considered correct only if it was performed within the benchmark. Note also the use of footnotes to indicate any discrepancies that occurred during the test. This provides a historical record without having to reference a second table or the raw data itself.

See the web site that accompanies the book (`www.wiley.com/go/usabilitytesting.com`) for additional examples of analysis spreadsheets and tallies.

Task Timings

Task timings relate to how much time participants require to complete each task. Common statistics that describe task timings include mean,

Module	Tasks	Percentage of participants performing correctly (within benchmark)	Mean time (minutes)	Standard deviation (minutes)
A	Set temperature pressure	83	3.21	4.38
	Set gas flows	33	12.08	10.15[1]
	Ignite the QRC	83	2.75	2.68
	Bake out column	100	0.46	0.17
	Set oven temperature program	66	6.54	2.56
	Run QRC checkout	100	0.83	0.34
	Program pressure and temperature	66	5.17	3.46
	Rerun checkout	100	0.29	0.09
B	Load the sample tray	100	0.88	0.28
	Create a subdirectory	33	8.00	3.92
	Create a sequence	66	9.42	7.70
	Start the sequence	66	1.00	0.92
	Stop the sequence	100	1.30	0.89
	Check the report	100[2]	1.38	0.58
	Reintegrate and print report	66	11.67	5.15
	Modify the method	100[3]	2.67	1.86
	Save the method	83	2.92	5.41
	Modify the sequence	66	3.46	3.96
	Restart the sequence	83	1.08	0.89

1 Participants 4's timings are not included. Participant had great resistance to performing the task and had not attempted it on the job in more than 18 months. Participant was also given misnformation during the task.

2 Participant 4 did not perform this task due to running the wrong method and having the test plot run.

3 Participant 1 did not perform this task to time constraints.

Figure 11-1 Performance score summaries and mean times ($N = 6$)

median, range, and standard deviation, each of which is explained in the following discussion.

- **Mean time to complete.** For each task, calculate the average time required by all participants to complete it, using the following formula:

$$\text{Mean} = \frac{\text{Sum of All Participants' Completion Times}}{\text{Number of Participants}}$$

  The mean time to complete is a rough indication of how the group performed as a whole. It can be compared to the original time benchmark developed for the task to see if users, in general, performed better or worse than expected. If it turns out that task times are very skewed either left or right — that is, the highest or lowest times are very different from the other times — consider using the median score, instead of the mean, as a comparison tool.

- **Median time to complete.** The median time to complete is the time that is exactly in the middle position when all the completion times are listed in ascending order. For example, if the times to complete a task (in minutes) for nine sessions were:

SESSION NUMBER	COMPLETION TIME (IN MINUTES)
1	2.0
2	2.3
3	2.3
4	3.0
5	**3.0**
6	3.2
7	3.8
8	4.0
9	16.0

Then the mean time to complete would be 4.4 minutes. However, the *median* time to complete would be 3.0 minutes (bold above), which because of the aberration in session #9, seems more typical of the scores.

- **Range (high and low) of completion times.** This shows the highest and lowest completion times for each task. This statistic can be very revealing if there is a huge difference between these two times. Especially for small sample sizes, where each participant's performance is crucial, you would like to know why there is such a huge difference between the high and low scores. Did the poorest performer view the task in an unusual way, one that a significant minority of the target population might share? Or, did that participant simply lack some important skills possessed by the target audience?

USING RANGE TO DETERMINE "OUTLIERS"

For performers who fall outside the range, the question is always if this is representative of the majority or if the participant lacked a skill. This sometimes happens, as it did in on particular test that Jeff did. The very last participant of eight performed much more poorly than the other participants on a task yet fit the user profile exactly. The design team was forced to decide if this poor performance was an aberration (sometimes called an "outlier") or if other end users might perform the task similarly.

- **Standard deviation (SD) of completion times.** The standard deviation, like the range, is a measure of variability — to what degree the times differ from each other. It reveals how closely the scores, in this case the completion times, are clustered around the mean time. Because the SD takes into consideration the middle as well as the end times, it is a more accurate indicator than simply using the longest and shortest completion times. The basic formula for calculating the standard deviation for a group of scores is:

$$\text{Standard Deviation (SD)} = \frac{\sqrt{\Sigma x^2 - \frac{(\Sigma x)^2}{n}}}{n-1}$$

Where Σ $\mathbf{x}^2$ the sum of the squares of each of the scores.

Σ $\mathbf{x}$ the sum of all the scores.

n the total number of scores.

As an example of a calculation, let's assume that you have the following series of four completion times, where x is the value of each time:

X (MINUTES)	X^2
6.0	36.00
5.5	30.25
4.0	16.00
5.0	25.00
Σ x = 20.5	Σ x^2 = 107.25

Then the standard deviation would be calculated as follows:

$$\text{SD} = \frac{\sqrt{107.25 - \frac{(20.5)^2}{4}}}{3} = \frac{\sqrt{107.25 - 105.06}}{3} = 0.73 \text{ minutes}$$

The SD is always stated in the original measurement units, in this case, minutes. Fortunately, the SD formula is included on almost any current computer spreadsheet program.

How can you use the SD? Suppose that you calculate a mean time of 6.0 minutes for a task, with an SD of 0.5 minutes. This represents a tightly clustered distribution around the mean, which implies that users performed very similarly to one another. If, however, the SD is 3 minutes with the same mean time of 6 minutes, that represents a much broader distribution of times. This much broader distribution could warrant a second look at users' performance to understand why they performed so differently from each other. Were there identifiable differences in experience that could cause this wide variation in scores? Or did some users simply miss an important piece of information?

Summarize Preference Data

In addition to summarizing performance data, you will also want to summarize the preference data that you collected. Preference data may come from multiple sources such as surveys, post-test questionnaires, and post-test debriefing sessions. Following are some guidelines for summarizing different types of preference data.

- **For limited-choice questions.** Sum the answers to each individual question, ranking, rating, and so forth, so that you can see how many participants selected each possible choice. You may also want to compute the average scores for each item, but for a small sample size this may not even be necessary in order to view trends.
- **For free-form questions and comments.** List all questions and group all similar answers into meaningful categories. For example, sum all positive and negative references to a particular screen or particular section of a document. This method will enable you to scan the results quickly for a general indication of the number of positive and negative comments.
- **For debriefing sessions.** Have all interviews transcribed, and pull out the critical comments. It can help to have interviews transcribed because it is so much easier to skim through written comments or search the electronic versions and pull out the essential information from a transcription than it is to listen and categorize the information while you are listening to the recording. In addition, a written record of these interviews makes the information much more accessible for later reference by others.

List the comments and observations from data collection sheets, both your own and those of participants. Organize these comments and observations in some workable form that makes sense for your test. For example, group them by task or by product component, such as a screen or section of a manual.

Figure 11-2 shows an example of a preference data summary. The summary includes the questions and choices that were provided to participants, as well as a summary of their responses.

Figure 11-3 provides another example of a data summary. The data in this example is even more compressed because, instead of including participants' comments, it shows only the number of times a participant referred to a particular product component or aspect. Especially for tests with more than 10 participants, this more compressed summary gives you a quick means of identifying those aspects that are of greatest concern and those that are of least concern.

Preference questionnaire

I thought the information I got from this web site was useful.

answer options	Strongly disagree	Disagree	Neither agree nor disagree	Agree	Strongly agree
count	0	0	1	6	5
percentage	0	0	8	50	42

What two things did you like about it? (As entered by participants themselves)

Candidate statement, and the opportunity to change my address and info.
Showed all locations for ballot drop off and easy to follow maps. Also showed information about all candidates current and running.
Information on upcoming elections and drop off places for the ballots.
Getting into the system easily and the fact that the forms are protected from copying.
Easy to navigate, Simple colors, concise, quick reference, learned things I didn't know, embedded map-handy!
The knowledge I gained from it. The ease of going through the web site.
Candidates' bios for upcoming election. Ease of changing address.
One–Stop Shopping. Intuitive.
Gave option to change address. Option to print registration from.
The candidate statements are available. I like that I can vote from home on the web.
Organized, easy to use.

I can do everything I would expect to be able to do on a web site for voters.

	Strongly disagree	Disagree	Neither agree nor disagree	Agree	Strongly agree
count	1	0	1	8	2
percentage	8	0	8	67	17

What two things are missing from the site?

Better descriptions of where the drop–off places are and the ability to click on candidates' names for info.
Like to see my current address so I don't have to change it to make sure its right.
Being able to vote online.
I'm not sure if anything I would need is missing.
I can't think of anything at this time. I need to spend more time checking out the web site.
Future election dates.
More in–depth information. Low Tech.
Candidate info. Broader view of district info.
Being able to look at other election going on in the city, past election results.

Figure 11-2 Usability survey data summary

By participant number

	email	IM	read news online
never			7, 8
once/ month		16	
once/ week		1, 11, 15	1, 15, 16
daily	1, 2, 3 8, 12, 15	2, 4, 7	2, 4, 5 6, 10, 11 12, 14, 16
several times/ day	4, 5, 6, 7, 9, 10, 11, 13, 14	3, 5, 6, 8, 9, 10, 12, 13, 14	9, 3, 13,

Number of responses

	email	IM	read news online
never			2
once/ month		1	
once/ week		3	3
daily	6	3	9
several times/ day	9	9	3

Figure 11-3 Compressed data summary

Compile and Summarize Other Measures

Above and beyond the standard set of descriptive statistics just discussed is a variety of other measures that may be of use. For example, you may want to note the following if appropriate for answering your research questions and the measures you planned for when you designed the usability test:

- Number of times returning to main navigation unnecessarily
- Number (and type) of hints or prompts
- Number of times the site map was accessed
- Points of hesitation (and for how long)

Compile and summarize these measures in your report, as needed, to diagnose problems and address test objectives.

Summarize Scores by Group or Version

If your test design included more than one user group, you will want to summarize the data separately for each distinct group to see whether one group is performing differently from the other. For example, Figure 11-4 shows a comparison of task accuracy scores (e.g., percentage of participants completing a task successfully) for two groups — novices and experienced users.

Percentage (and number) of participants completing a task successfully

Group	Task 1	Task 2	Task 3
Novice users	83% (10)	50% (6)	66% (8)
Experienced users	66% (8)	100% (12)	100% (12)

Figure 11-4 Task completeness by group

Similarly, if you tested different versions of a product or materials, you should compile summaries of performance on each different version. For example, Figure 11-5 shows a comparison of the number of errors made by participants on three different versions of a manual.

Figure 11-6 shows a more comprehensive comparison summary from an exploratory usability test that compared two product prototypes, one a ''radio button'' type interface, prototype A and the other a graphic representation of the product, prototype B.

Number of errors by version

Group	Version A	Version B	Version C
Novice users	6	6	15
Experienced users	9	20	12

Figure 11-5 Number of errors, compiled by version

Summary of performance and preference

Participant	Group	Version A # tasks correct	Version B # tasks correct	Liked best	Prefer to teach a novice	Version A Ease of use (1–5)	Version B Ease of use (1–5)
P1	N	*12/15	11/15	A	B	4	3
P2	N	12/15	*11/15	B	B	3	5
P3	N	*12/15	10/15	A	B	5	2
P4	N	10/15	*13/15	B	B	2	4
P5	E	*11/15	10/15	B	A	4	3
P6	E	11/15	*13/15	B	A	5	4
P7	E	*12/15	10/15	B	B	3	4
P8	E	13/15	*11/15	B	B	4	3
P9	N	*9/15	13/15	A	A	4	3
P10	N	12/15	*11/15	A	B	3	4
P11	E	*11/15	13/15	B	B	4	4
P12	E	14/15	*12/15	B	B	4	4
					Avg.	3.8	3.6

Key:
N = Novice
E = Experienced
* = Participant saw this version first

Figure 11-6 Summary of performance and preference rankings

One note of caution about using percentages with small sample sizes: If you only have 8 or 10 participants in your study, talking in terms of percentages can sometimes seem to inflate an effect. For example, an 83 percent success rate sounds like a big deal. But saying that 10 of 12 participants completed tasks on prototype A grounds the finding in reality.

Analyze Data

After you have transformed the raw data into more usable summaries, it is time to make sense of the whole thing. Please note that the decision to summarize by task was a deliberate one, because the task represents the viewpoint or goal of the users and what they would like to achieve. Staying task-oriented while summarizing and analyzing the data forces you to look at the situation from the viewpoint of the users, which is the ultimate reason for testing. Can the users perform their tasks using your product? If they cannot, you then need to determine which component, or combination of components, is the culprit — and to what extent. To begin the analysis, identify the tasks on which users had the most difficulty. That will keep you focused on the worst problems.

Identify Tasks That Did *Not* Meet the Success Criterion

Going into your study, your team defined ''success'' for each task (see the section ''A Description of Successful Completion of the Task'' in Chapter 5), by specifying criteria for proper completion. Simply put, a task that does not meet a criterion is one that a predetermined percentage of participants did not complete successfully within a specified benchmark, if one was specified. As mentioned in Chapter 3, we have come to use a 70 percent success criterion for a typical assessment test. If at least 70 percent of participants do not successfully complete a task, then it is ''difficult'' or ''problematic.'' It then gets attention in analysis and reporting. Essentially, such tasks represent the vulnerable portion of the product and its support materials. If you do a preliminary report, tasks that participants had difficulty with or completely failed to be able to do are the ones on which you should focus first.

The 70 percent criterion represents a reasonable balance between being too demanding and too lax, especially in a test of an early version of a product. Assuming that you are doing small tests and iterating design in between them, as the tests go on, the likelihood of reaching (and surpassing) the 70 percent success criterion should grow. Eventually, you would like users to approach a 95 percent success rate, but if you demand that for the first usability test in a planned series of tests, you will often flag almost all tasks. The difference between 95 percent and 70 percent is what the design team has to bridge by its improvements to the product. Conversely, if you make the criterion too low, such as 50 percent, then too many product deficiencies are being assigned to a lower priority.

In Figure 11-7, the eight non-criterion tasks are shaded, that is, the eight tasks from the data summary (originally shown in Figure 11-1) that have success rates below 70 percent. If this data had been collected during a validation test, you would flag these eight tasks in exactly the manner shown in Figure 11-7.

While the 70 percent criterion rule works quite well for providing a snapshot of problem areas, you may have a completely different method for identifying the most difficult tasks. For a test with few participants, for example, the most difficult tasks will usually jump right out at you, without need for much analysis. However you do it, you must make a distinction between levels of performance, rather than just listing and lumping all the results together into one long table. Distinguishing between different performance levels allows you to focus on the problem areas.

Module	Tasks	Percentage of participants performing correctly (within benchmark)	Mean time (minutes)	Standard deviation (minutes)
	Set temperature pressure	83	3.21	4.38
	Set gas flows	33	12.08	10.15[1]
	Ignite the QRC	83	2.75	2.68
	Bake out column	100	0.46	0.17
A	Set oven temperature program	66	6.54	2.56
	Run QRC checkout	100	0.83	0.34
	Program pressure and temperature	66	5.17	3.46
	Rerun checkout	100	0.29	0.09
	Load the sample tray	100	0.88	0.28
	Create a subdirectory	33	8.00	3.92
	Create a sequence	66	9.42	7.70
	Start the sequence	66	1.00	0.92
	Stop the sequence	100	1.30	0.89
B	Check the report	100[2]	1.38	0.58
	Reintegrate and print report	66	11.67	5.15
	Modify the method	100[3]	2.67	1.86
	Save the method	83	2.92	5.41
	Modify the sequence	66	3.46	3.96
	Restart the sequence	83	1.08	0.89

1 Participants 4's timings are not included. Participant had great resistance to performing the task and had not attempted it on the job in more than 18 months. Participant was also given misnformation during the task.

2 Participant 4 did not perform this task due to running the wrong method and having the test plot run.

3 Participant 1 did not perform this task to time constraints.

Figure 11-7 Non-criterion tasks shaded

Identify User Errors and Difficulties

After you highlight the non-criterion tasks, identify the errors that caused the incorrect performance. An error in this case is defined as any divergence by a user from an expected behavior. It is very helpful if you can define what an error is before conducting the study. You must do this in a validating or summative test. (This is where doing one or more pilot sessions can be extremely useful.) The purpose of an exploratory or formative test may be to understand what the possible errors are. For example, the user was supposed to enter a customer ID in field 10, but instead entered it into field 11. Or, the user was supposed to delete a backup file, but instead deleted a working file. Or, the user simply omitted a step. These all represent errors that resulted in unsuccessful completion of a task.

Conduct a Source of Error Analysis

Now the real fun begins. Identify the source of every error, if possible, by noting the responsible component or combination of components, or some other cause. *This is your transition point from task orientation to product orientation*. This type of analysis is the ultimate detective work and is the most labor-intensive portion of the post-test regimen. Your objective is to attribute a product-related reason for user difficulties and/or poor performance. Essentially, one has to be clear about why user errors occurred, otherwise the recommendations cannot be accurate. Therefore, take your time and do a thorough job here. This is the point at which you might go back and review the recordings. But don't just go off and do this by yourself. You can use this analysis as a framework for workshops with others who had observed the test. Together, you can review the data in light of who the participants were. Together, you can agree on what the sources of problems were.

Of course, some sources of error will be *dead-on* obvious, and will take very little probing. For example, if a user's task was to enter a 20-character customer record identifier within a field on a screen and the field was only long enough to contain 11 characters, the reason for the problem is obvious.

Other determinations will be much more challenging. For example, if customers of a health insurance company were unable to perform a complex decision making process that took them through three sections of web application, four different sections of a benefits handbook, and two online chats with a customer service rep, identifying the source of error is an order of magnitude more difficult. Each of the aforementioned components probably contributed in its own way to the problem. However, there is usually a primary and a secondary culprit, and you should so note. For example, when multiple components are involved, confusing navigation may be the primary problem, followed by the content of the documents, and last by the information from the service rep. Clearing up the primary problem invariably simplifies the changes required to the secondary sources.

To perform the source of error analysis, you have many areas to review and consider. You have your notes and memory, the notes and memories of others, your understanding of how the product works, possibly the video recordings, and, equally as important, your understanding of user-centered design. You also need to consider the background of the users who made errors.

For particularly challenging or critical errors, such as the situation described above, go back to any data collected by monitoring software and/or review the video recordings of several users who erred. You may have missed something important during the actual test.

Try not to solve the problem prematurely and recommend a fix before you have identified *all* the sources of error. In terms of thoroughness, you should ideally perform a source of error analysis for every task performed by every user. In this way, you will account for every deficiency and recommend action for each one as well. Figure 11-8 shows an excerpt from a source of error analysis. Note that there are two sources of error: the flow of the transaction (which did not match how the users thought of the task), and the information architecture (which used domain-specific language unfamiliar to users).

Task	Source of error
Use points to reserve a hotel room	• Nothing in the web site said users must be logged in before starting the reservation process to be able to use points. • Participants expected to be able to sign in to see how many points they had after Step 3 (Check availability), but there was no way to log in there. • After the home page, the search widget that included an option to redeem points was in the lower–right corner of the page, and none of the participants noticed it. • Participants looked for a link to FAQs; they should have selected the Customer Service link, but participants inferred that the Customer Service link would lead to a phone number or online chat.

Figure 11-8 Excerpt from source of error analysis

Prioritize Problems

There are many ways to rank usability problems. Criticality is one. After you identify the specific sources of errors, the next step is to prioritize these problems by *criticality*. Criticality is defined as the combination of the severity of a problem and the probability that the problem will occur. If you represented criticality as an equation, it would look like this:

$$\text{Criticality} = \text{Severity} + \text{Probability of Occurrence}$$

The reason for prioritizing problems by criticality is to enable the development team to structure and prioritize the work that is required to improve the product. Obviously, you want the development team to work on the most critical problems first, assuming that there is time prior to the next release. Here is one way to prioritize problems by criticality.

First, categorize a problem by severity. We measure severity on a four-point scale, with each problem ranked. Figure 11-9 shows one severity rating system; Figure 11-10 shows another.

Severity ranking	Severity description	Severity definition
4	Unusable	The user either is not able to or will not want to use a particular part of the product because of the way that the product has been designed and implemented. Example: Product crashes unexpectedly whenever it is powered on at altitude.
3	Severe	The user will probably use or attempt to use the product, but will be severely limited in his or her ability to do so. The user will have great difficulty in working around the problem. Example: Synchronizing the device to another device can only happen when certain files are not in use. It isn't obvious when the files are in use.
2	Moderate	The user will be able to use the product in most cases, but will have to take some moderate effort in getting around the problem. Example: The user can make sure that all complementary applications are closed while syncing the two devices.
1	Irritant	The problem occurs only intermittently, can be circumvented easily, or is dependent on a standard that is outside the product's boundaries. Could also be a cosmetic problem. Example: The message area of the device's small screen is at the very top, dark blue, and often shaded by the frame of the screen.

Figure 11-9 Problem severity ranking

4 = Task failure–prevents this user going further

3 = Serious problem–may hinder this user

2 = Minor hindrance–possible issue, but probably will not hinder this user

1 = No problem–satisfies the benchmark

Figure 11-10 Problem severity ranking

Frequency ranking	Estimated frequency of occurrence
4	Will occur ≥90% of the time the product is used
3	Will occur 51–89% of the time
2	Will occur 11–50% of the time
1	Will occur ≤10% of the time

Figure 11-11 Frequency of occurrence ranking

Next, rank the problem by estimated frequency of occurrence. That is, estimate the probability that a problem will occur in the field, and convert that estimate into a frequency ranking. See Figure 11-11 for a table of frequency rankings. (Don't get nervous if this is hard — this is only an estimate.)

To arrive at your estimated frequency of occurrence, you need to account for two factors:

- The percentage of total users affected
- The probability that a user from that affected group will experience the problem.

Therefore, if you feel that 10 percent of the target population will encounter this problem about 50 percent of the time, then there is only a 5 percent estimated frequency of occurrence ($0.10 \times 0.5 = 0.05 = 5\%$). Do not worry about the exact precision for your estimated frequency of occurrence. Your best guess will still be quite meaningful.

Ascertaining a problem's criticality is then a simple matter of adding the severity ranking and the frequency ranking for that problem. For example, if a particular problem is ranked unusable (severity ranking = 4), but will only occur 5 percent of the time (frequency ranking = 1), then that problem would receive a priority of 5 (4 + 1). Similarly, if there were a problem that was simply an irritation but affected almost *everybody*, then that would also get a 5. Using this method, the very highest priority are assigned to problems that made the product unusable for everyone. These priorities can then help you and the development team decide how to focus resources, concentrating on fixing the more critical problems first if there is a time constraint. In an ideal situation, every problem would be fixed before release, but that rarely happens. Keep in mind that you can develop your own hierarchy and definitions for criticality based on your organization's objectives and the particular product you are testing.

For simple tests, there is an easier way to ascertain which problems are most critical and in most need of attention. *That is to simply ask participants during the debriefing session to tell you what was the most problematic situation for them*. If you find that several participants are in agreement about priorities, then that is an important indication about where to focus your resources.

Analyze Differences between Groups or Product Versions

If you have conducted a comparison test, you may want to compare the differences between groups or between different versions of your product. For example, you might compare the difference between the use of an old versus a new checkout sequence for an e-commerce web site or the difference between the performance of novice and experienced users.

You will do this by analyzing the amount, types, and severity of errors that users made for the two (or more) versions or groups, as well as users' preference ratings, rankings, and general comments. This analysis can be very challenging especially if there is no clear-cut ''winner,'' as in the following example.

Figure 11-12 summarizes the results of a comparison test of two prototypes (previously compiled in Figure 11-7). A simple review of the number of errors and the ease-of-use rankings reveals that the two prototypes are very close in user performance and preference. There is a slight advantage in performance for the A prototype and a clear advantage in preference for the B prototype. That much is clear.

What is not clear and what can only be deciphered from notes and observations are the *types* of errors that participants made, the assumptions they made even when they did perform correctly, and what they said about using the product. As it turns out, the reason that the B prototype did not perform as well as the A prototype was that it was unfamiliar to this group of participants. The participants were much more familiar with the interaction elements on A, which will also be true for the intended customers.

However, the B prototype resulted in more positive comments, especially as users mastered its subtleties. It also seemed more intuitive for new users. Almost all of the participants said they would prefer to teach the graphic interface to a novice, due to its particular representation of the product. Participants felt that novices could see relationships of different parts of the overall system on the B interface, while these relationships were only inferred on the A interface.

Lastly, the source of error analysis was very revealing. Most of the B prototype errors occurred because of the poor quality of the visual elements. The participants simply misinterpreted the graphic representations of system parts, such as ports and buttons. Had the graphics been more realistic, the error rate might have been nil.

For all these reasons, the decision was made to move to a graphic interface such as the one in the B prototype for the initial top-level screen, with radio buttons used for lower-level selection screens, such as those used in the A prototype.

Summary of performance and preference

Participant	Group	Version A # tasks correct	Version B # tasks correct	Liked best	Prefer to teach a novice	Version A Ease of use (1–5)	Version B Ease of use (1–5)
P1	N	*12/15	11/15	A	B	4	3
P2	N	12/15	*11/15	B	B	3	5
P3	N	*12/15	10/15	A	B	5	2
P4	N	10/15	*13/15	B	B	2	4
P5	E	*11/15	10/15	B	A	4	3
P6	E	11/15	*13/15	B	A	5	4
P7	E	*12/15	10/15	B	B	3	4
P8	E	13/15	*11/15	B	B	4	3
P9	N	*9/15	13/15	A	A	4	3
P10	N	12/15	*11/15	A	B	3	4
P11	E	*11/15	13/15	B	B	4	4
P12	E	14/15	*12/15	B	B	4	4
					Avg.	3.8	3.6

Key:
N = Novice
E = Experienced
* = Participant saw this version first

Figure 11-12 Summary of performance and preference rankings

The previous example illustrates how such comparisons usually work. There are many types of data to analyze and sift through before solid conclusions and recommendations can be formulated. Typically, the source of error analysis is especially important for comparisons. This is because each version usually has distinct advantages and disadvantages. Only by understanding the types and sources of errors in depth is it possible to ascertain the best version. Because neither of the versions being compared is a clear winner, that is, neither ever has all of the advantages or all of the disadvantages, a comparison test inevitably results in a hybrid version that incorporates the best elements of the prototypes.

Using Inferential Statistics

To this point, you have analyzed the test data using simple descriptive statistics. For example, the means, medians, and ranges of times all *describe* the characteristics of your data in ways that help to see patterns of performance and preference and ascertain usability problems. You have also reviewed the strictly qualitative data such as specific comments made by participants. For the vast majority of tests that we conduct, and for the vast majority of readers of this book, such analysis is sufficient to make meaningful recommendations.

Occasionally though, the development team or whoever has commissioned the usability research may insist that one obtain statistically significant results. Most often, this situation arises when two versions of a product are being compared with each other and much is riding on the outcome. To obtain statistically significant results requires the use of inferential statistics. That is, we *infer* something about a larger population from the smaller sample of test participants. If the results of a test are statistically significant, then you can assume that if you conducted the study over again with different people with similar experience and background, you would get the same results. However, the use of inferential statistics opens a huge can of worms, and we recommend extreme caution.

First of all, many of those involved in usability tests have not been sufficiently trained in the use and interpretation of inferential statistics. Even among seasoned professionals, there can often be much disagreement about exactly which statistical test to use and what the results imply afterward. Deciding which statistical technique to use is not trivial and depends on the following factors:

- The scale or measurement used for the conditions (variables) being tested
- The number of conditions and the number of levels in each condition
- The number of conditions that will be analyzed at the same time
- The way in which participants were assigned to groups
- What you will infer from the statistical method

Second, those using the results of the test to make decisions about the product are rarely trained in interpreting such statistics, and can easily misinterpret the results. It is important that they receive an explanation of what the statistical results ''prove'' and ''disprove.''

Third, and probably most relevant, the way in which you conduct the test will vary greatly depending on whether you are trying to obtain statistical results or not. Obtaining statistical results requires a more rigorous design and probably a fairly large sample of participants.

For example, when comparing two or more versions of a product, you must have rigorously controlled the conditions that differ in each version during the test. If you were testing to see if the *format* of a new version of a document was easier to use than the format of an old version, but in addition to a revised format, the new document had additional content and an improved index, you would find it very hard to isolate the effects of format on performance. To support a hypothesis that one format will result in improved performance, the versions should differ in *format* alone and nothing else.

Also, if you conduct a test with much probing and interaction between test moderator and participants, then it is very easy to bias the results of one of the versions to the detriment of the other for the purpose of proving a hypothesis.

Your sample size is also crucial. If your sample size is small, as in the previous example of the two prototypes (see Figure 11-12), you will have difficulty obtaining statistical proof that the results were not due to chance. As a very general rule, all things being equal, you should have sample sizes of at least 10 to 12 participants per condition before considering the use of inferential statistics.

To summarize, the appropriate use of inferential statistics is both a complex and subtle topic. While there certainly is a place for their use in usability research, only those with a thorough grounding in experimental design and statistical theory make use of this tool. For the vast majority of practitioners, we suggest avoiding the use of such statistical techniques for the reasons mentioned previously. If you would like to learn more, we recommend reading one of the many introductory books on probability and statistics, such as *Statistics for People Who (Think They) Hate Statistics* by Neil J. Salkind (Sage Publications, 2007) or *Statistics in Plain English* by Timothy C. Urdan (Erlbaum, 2005). Depending on your study goals, you may want statistics oriented to behavioral sciences, social sciences, or market research — or something else.

An even better plan, if you are genuinely interested in learning more, is to enroll in a university course on statistics offered through either the university's social science or behavioral sciences department.