Building Shiny Apps the BOAST Way

A How To and Official Style Guide

Neil Hatfield (njh5464@psu.edu) and Robert Carey (rpc5102@psu.edu)

20 November 2020

Contents

Welcome

One of the biggest advantages of the modern era is computing power. This provides with the ability to present statistical concepts in ways that allow students to explore and build their understandings in dynamic ways. A growing tool on this front is the advent of applications built in the `R` language and utilizing the `shiny` package by the RStudio company; these are called "Shiny Apps".

We are not the first group to build online applications to help students learn ideas, nor will we be the last. In fact, we're not even the first group to build Shiny apps for Statistics Education. Thus, there are many different approaches to how we go about designing and coding our Shiny apps. The primary goal of this document is to layout the approach that we use to 1) formalize the approach for ourselves, and 2) share our approach with others who are wanting to build their own Shiny apps for teaching.

This guide spells out the recommended approach for building and the required styling to should be used for all apps included in the Book of Apps for Statistics Teaching (BOAST). You will need to follow this Style Guide to ensure that any app you create meets (or exceeds) our standards is worth of inclusion in BOAST.

Effective Date: [to be listed here]
All apps created and/or modified after the Effective Date are required to adhere to this version of the Style Guide. Apps which have not been modified since before the Effective Date will be brought into compliance on rolling basis.

0.1 Organization

We've laid out this document into several parts, each with their own chapters.

- Part 1: Getting Started
  - Chapter 1—Explain Your Idea

  - Chapter 2—Using a Workflow

- Part 2: Getting Ready to Code

  - Chapter 3—Setting Up Your Computer

  - Chapter 4—Exploring the Boast Template
- Part 3: Style Guide–Coding
  - Chapter 5—BOAST Coding Style
  - Chapter 6—HTML and CSS
  - Chapter 7—Additional Coding Information
- Part 4: Style Guide–App Layout
  - Chapter 8—General Layout
  - Chapter 9—Common/Reoccurring Elements
- Part 5: Style Guide–Visual Style
  - Chapter 10—Color
  - Chapter 11—Static Images
  - Chapter 12—R Plots
- Part 6: Style Guide–Language
  - Chapter 13–Wording
  - Chapter 14–Documentation
- Part 7: Accessibility and Mobile Devices
  - Chapter 15–Accessibility
  - Chapter 16–Mobile Friendliness

Yes, this looks long but keep in mind the following:

> Building an app is easy. Building an app that someone else can use is a challenge.

> While building an app that supports a user in developing a procedural conception of an idea is easy, building an equitable app that supports anyone in developing productive meanings is more difficult.

Our mission with BOAST is to achieve both with our entire collection. Thus, writing downs all of the ins and outs of our process will take a significant amount of space. Whenever possible, we've attempted to provide you with examples to help you see our standards in action as well as code that you can use as templates. For each code block, you should be able to place your cursor over the block and see a copy icon in the upper-right hand corner. This will allow you to copy the code. We've also included links to additional resources so that you can learn more about key topics.

Keep in mind that this is a living document. As issues arise, we will update this guide to address them. Additionally, as we think of potential issues, we'll also update the guide to provide guidance before they occur. We also welcome any suggestions for improvements.

0.2 Why Our Approach?

While we could go on about the many reasons for our approach, we will focus on the critical ones:

0.2.1 We're a Team of Developers

From the start, the Book of Apps for Statistics Teaching (BOAST) has been the joint effort of faculty members and students. Having a clear articulation of how to make apps within our environment is critically to ensure that we have cohesiveness between the apps and through the years as the team changes. Our approach is meant to maximize the team's ability to adapt, maintain, and grow BOAST throughout the years.

0.2.2 Research Driven

We place a premium on drawing from what researchers have found about student thinking on various topics. And for those topics where the literature is thin or non-existent, we can fall back on the wealth of experience and expertise our faculty have. When we design apps, we focus on what the learning objectives are and work on how we might support students in developing meanings that are consistent with our targets.

0.2.3 Accesiblity Minded

One of the key ways in which our approach differs from a vast majority of others is the level of accessibility thinking we go through. Shiny apps are **not** very accessible by default and are extremely easy to make even more inaccessible.

For example, consider the following entries in the 2020 Shiny app competition, as of 10/15/2020:

- Blog Explorer by Stefan Schliebs: This Grand Prize winner has 11 errors, 68 contrast errors, and 14 alerts.
- Life of Pi: A Monte Carlo Simulation by Zauad Shahereer Abeer: This Honorable mention has 17 errors, 13 contrast errors, and 9 alerts.
- OTS Beta Dashboard by Mauricio Vargas: This Honorable mention generates 14 errors, 0 contrast errors, and 5 alerts.
- Probability of Normal Distribution by Aep Hidayatuloh: Produces 5 errors, 108 contrast errors, and 27 alerts.
- Didactic Modeling Process: Linear Regression by Daniel Rivera: while not part of the Shiny contest, this is a currently featured app in RStudio's Shiny User Showcase, which has 57 errors, 26 contrast errors, and 184 alerts.

We used WebAIM's Web Accessibility Evaluation Tool (WAVE) to get these measurements. However, we must point out that the above numbers are *under*-estimates. The nature of a Shiny app interferes with WAVE's scanning tool (for example, non-displayed app pages aren't checked) as well as the fact that rendered R plots can only be checked for the presence of alt text/aria labels but not contrast errors, small font sizes, etc.

Now, this isn't to say that our problem free. However, our process ensures that the number of errors that exist are in the single digits. When we publish an app, most of the remaining errors are ones we inherent from the `shiny` package and we're striving to address.

0.2.4 Avoid the "Shiny" Problem

In Education, and particularly in the American system, we wrestle with the "silver bullet problem". This problem is the belief that some reform effort will function as a fix or cure-all for the issues facing the education system. These reforms never bare out.

In design, people can get caught up with new (at least to them) tools such as a glitzy new system to do something. They will then start incorporating it everywhere they can. However, they often don't stop to think about whether or not this glittery new toy is appropriate or sound design. This is especially true for novice designers.

When dealing with Shiny apps for teaching, we face both problems simultaneously, which we call "The Shiny Problem". We work hard to ensure that

1. Our apps aren't going to be viewed as silver bullets but as useful resources that can supplement instruction
2. Just because `R` can do something doesn't automatically mean that we're going to include it; there must be a sound educational and pedagogical reason for including it.
3. Just because there's a fancy new package that's been released doesn't mean we'll blindly adopt it and use it wherever; there must be a sound educational and pedagogical reason
4. Just because someone else made an app that did such-and-such doesn't we mean we're going to; there must be a sound educational and pedagogical reason.

A key exception to the above deals with coding and app performance, but these are secondary to sound education and pedagogical reasons. We strive for our Shiny apps to not become *shiny*.

Part I

Getting Started

Chapter 1

Making an App

Sooner or later, you'll reach the point where it is time to make your own Shiny App for BOAST. Approaching your App's construction in a systematic way will help you tremendously. The following suggested workflow is an abbreviated version of the one in Section **??**:

1. Read the Style Guide
2. Identify a topic
3. Sketch out your plans
4. Create a new repository on GitHub
5. Begin writing the code
6. Edit and locally Test your code
7. Push your code to GitHub/Create a dev branch
8. Push your edits
9. Larger scale testing
10. Pull Request
11. Additional tweaks

This workflow (both in the more expanded state and in the abbreviated one) leaves out a lot of the going-ons and decision making that goes into making an app. Our goal with this section is to walk you through this process as Neil makes an app from scratch.

1.1 Step 1: Read the Style Guide

This is step is critical to making an app that is easy to debug (vital when programming) and is in-alignment with the Standards that we have set for BOAST. Please take your time going through the Style Guide and keep an eye on it for updates. You can access the published Bookdown version by clicking on the address link located at the top of the Guide's GitHub Repository.

As you build your App, we recommend periodically checking the Style Guide to 1) ensure that you're still adhering to it, 2) to stay up-to-date with the Guide, and 3) to see (and then use) any useful code chunks that are listed. Even as the authors of this Guide, we're constantly looking things up. Thus, there is no shame in constantly referring to the Style Guide; after all, it is meant to be a Go-To Reference.

1.2 Step 2: Identify a Topic

This is one of the more challenging tasks for anyone: deciding what to make an app about. The notion of "picking a topic" is a bit misleading as not only should you pick a topic, but you also need to pick the purpose of the app, and the goals. While all three go hand-in-hand with each other, the easiest decision of the three is purpose.

1.2.1 Your App's Purpose

When we look across BOAST, we can categorize the apps into two major classes: apps whose purpose is to help students **learn** an idea/concept and apps whose purpose is to help students **review** an idea/concept. The NHST Caveats App stands as a good example of an app whose purpose is to help students learn something. On the other hand, the [Null] Hypothesis Testing Game (Tic-tac-toe) and the Matching Distributions are examples where the student is reviewing their understandings. There are some apps such as ANCOVA that are dual purpose, having both learning elements and reviewing elements.

Generally speaking, apps whose purpose is **learning** are going have Explorations and possibly Challenges for Activity Tabs. Apps whose purpose is **reviewing** will have just Games for Activity Tab(s).

By identifying early on the purpose of your App, you'll be able to better plan your app.

1.2.2 The Topic

Here is where you'll decide what statistical idea/concept your App is going to be centered around. While you should be as specific as possible here, **apps for learning** need to have much more specificity than **apps for reviewing**. Saying that you want your App "to deal with probability" is not going to be as useful as saying that you want your App "to help students recognize when probability is and is not appropriate for different contexts".

Don't feel like you have to details to the nanoscale. As you continue developing and talking with people, you'll start adding in more details.

You can identify potential topics through any of the following:

- Think of a concept in a Stat class that you (or your peers) struggled with (or still do)
- Think of a concept where you thought to yourself "I wish there was a way to visualize this"
- Think of a concept where you went "How does this work?"
- Did you see/hear about a new study and thought to yourself "what's going on?"
- Did a media report make you do a double take?
- Ever think to yourself "I wish I had more practice with [topic]."
- And more

Again, this list is just a starting place to identify potential topics. Draw inspiration from your experiences, your curiosity. Bounce ideas off of each other and the faculty.

Word of Caution #1: While it is okay to get ideas from other websites, please keep track of this. We need to give credit where credit is due. If you come across an app (whether in Shiny or some other program) somewhere and you think that it would be good to translate that into Shiny app for BOAST, we need to know.

Word of Caution #2: An unproductive place to begin is to say "Hey, I want to make a memory game" or "I saw this really cool animation and I want to make an app that does that." Both of these (especially the second) place the educational aspect of the app as a secondary focus. **This is not consistent with BOAST.** All apps should place educational purposes and goals before anything else. Forcing a topic to make use of certain tool because that tool would be "cool" or "fun" does not further our goals. We should strive for synergy between topics and the tools within Shiny.

1.2.3 Goals

Now, we're sure that some people have already thought to themselves "hey, you've already had us identify purpose so isn't this the same?" To which we reply "No." We've used **purpose** to refer an overarching aim of your App. Here, **goal** refers to a much more contextualized aspect. While these goals will be less important for reviewing apps, you will need to identify specific learning goals for learning apps.

For this step, you're going to need to work with the faculty to identify the learning goals for your app. If you're drawing from a past experience in a course, look to see if there were learning objectives listed. Those can be extremely useful starting places.

1.2.4 Example

The following is a somewhat stream-of-consciousness narrative of how Neil tackled this step. We'll place flags in parentheses.

I would like to make an app that would help introductory students (*identifies an intended audience*) wrestling with some of the concepts of descriptive/incisive statistics (*Purpose: learning*). Some ideas that I have are:

- Conceptualizing the two distinct uses of the *sample arithmetic mean*–mitigation of measurement errors of a single object vs. measuring how well a group of objects/beings performs
- Conceptualizing statistics as functions that measure attributes of collections
- Conceptualizing the relationship between the values of the *sample median* and *sample arithmetic mean* is not a competition (i.e., one is not "better than" the other)
- Avoid falling into the skewness trap for the ordering of the values of the *sample median* and *sample arithmetic mean*.

This last option feels like a fairly straightforward (and simple) app to create. Additionally, there's an article by von Hippel on this issue I can reference. (*Topic: relationship between the values of* sample median *and* sample arithmetic mean *in the presence of skewness*)

After using the app, a student should recognize that while for a many data collections that have skewness, the value of the *sample arithmetic mean* will be either greater (for positive skew) or less (for negative skew) than the value of the *sample median*, this is not an absolute rule. (*Learning Goal*)

1.3 Step 3: Sketch out plans

Something to keep in mind with this workflow is that it is not strictly linear. That is, just because you have moved into Step 3, doesn't mean that you can't go back to Step 2 and make revisions. Additionally, as you get into Steps 5-8, you might find yourself coming back to Step 3 (and possibly Step 2) and making changes. This said, sketching out your plans will help you as you move forward with developing.

There are several things that you'll want to be sure that you include in your sketch:

- A Suggested Title
  - This might not be what you settle on, but you'll need to have this firmed up by the time you get to Step 4.
  - You'll want to have both a [Long/Formal] Name and a shortened name ready
- Goal(s) of the App
  - Carry this forward from Step 2 as a tangible reminder
  - If you have several goals, you might consider denoting which tabs go with which goals
- What will the user be doing

  - Thinking about what you want the user to do
  - Identify anything the user might click on, drag, or otherwise manipulate.
  - If there are going to be options, identify what those options should be.
  - Will there be different levels?
  - Will there be a Challenge tab?
- What will the user experience
  - Think about the general layout (explore, game, etc.)
  - What outputs should the user be examining
- What are the relationships between elements
  - Think about how each input impacts the outputs
  - If you want the user to have a certain experience, does the coding allow for that experience to happen?
- Other Elements
  - Are you going to reference any external images?
  - Are you going to import any data files? If so, which ones? Do you have them?
  - Is there an activity packet?

The format of your sketches is up to you.

1.3.1 Example

As I start thinking about what this app might look like, I'm going to use a sheet of paper in quarters to organize some thoughts. Figure **??** shows the entire sheet of paper. The quadrants move left to right, top to bottom.

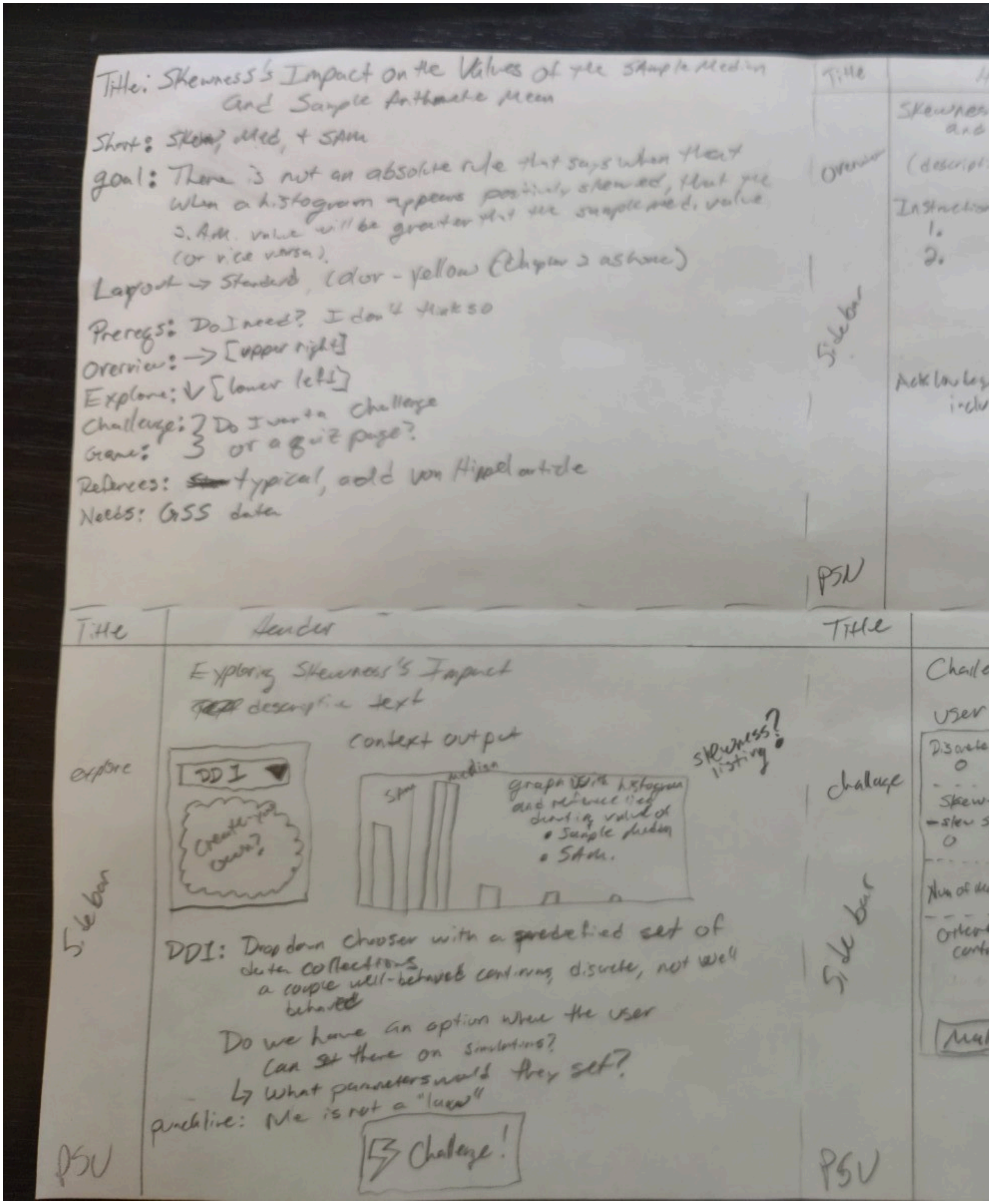
In the first quadrant (Figure **??**), I've listed a title ("Skewness's Impact on the Values of the *Sample Median* and *Sample Arithmetic Mean*"), a possible short title ("Skew, Med, SAM") as well as a restatement of my goal. I've then listed general aspects about the layout of my app, including what skin/theme color (yellow) I would need to use.

I've additionally identified that I need to include the von Hippel article in my Reference Tab as well as locating a version of the General Social Survey (GSS).

The second quadrant of my paper is the Overview Tab sketch (Figure **??**. I don't have much information here.

The next quadrant (Figure **??**) is where I started detailing the first Activity Tab of my app: the first Exploration. I'm imagining that in this exploration that the user would look through a variety of cases where the "rule" does and does not work. What I want the user to walk away from this tab is that just because a histogram looks positively skewed does not imply that the value of the *sample arithmetic mean* is greater than the value of the *sample median.*

I'm imagining a drop-down menu that would allow the user to move through a variety of data sets. Ideally, these data sets would be real data. The General



Title: Skewness's Impact on the Values of the Sample Median and Sample Arithmetic Mean

Short: Skew, Med, + SAM

goal: There is not an absolute rule that says when that when a histogram appears positively skewed, that the s. A.M. value will be greater that the sample med. value (or vice versa).

Layout → Standard, color - yellow (chapter 2 as home)

Prereqs: Do I need? I don't think so

Overview: → [upper right]

Explore: ↓ [lower left]

Challenge: } Do I want a challenge
Game: } or a quiz page?

Refences: ~~Sta~~ typical, add von Hippel article

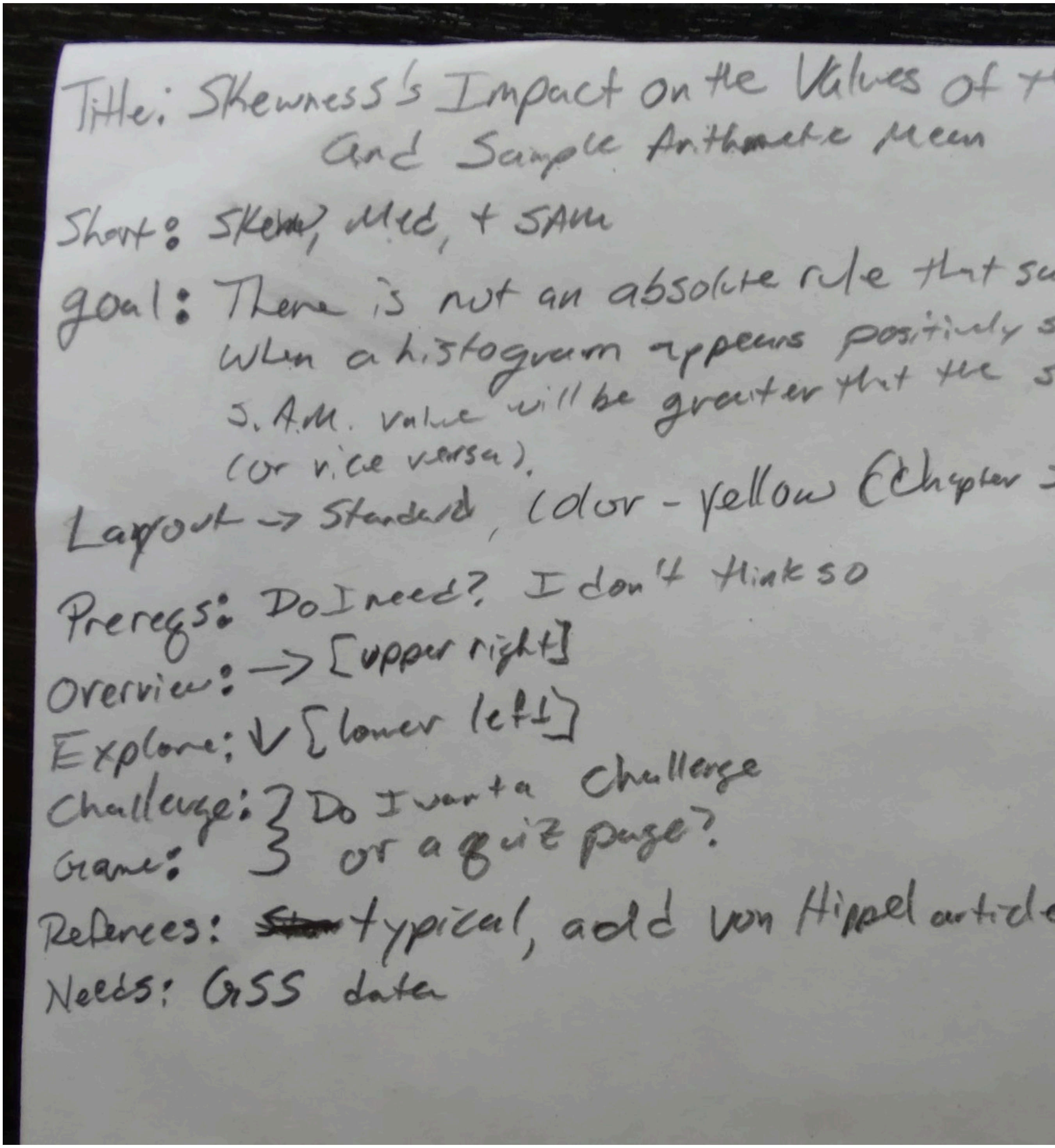
Needs: GSS data

DDI: Drop down chooser with a predefined set of data collections
a couple well-behaved continuous discrete, not well behaved

Do we have an option where the user can set there on simulations?
↳ What parameters would they set?

punchline: Me is not a "law"

Figure 1.1: Picture of My Plan



Title: Skewness's Impact on the Values of t
And Sample Arithmetic Mean

Short: Skew, Med, + SAM

goal: There is not an absolute rule that su
when a histogram appears positively s
S.A.M. value will be greater that the s
(or vice versa).

Layout -> Standard, color - yellow (chapter =

Prereqs: Do I need? I don't think so

Overview: -> [upper right]

Explore: ↓ [lower left]

Challenge: } Do I want a challenge
Game: } or a quiz page?

References: ~~Stan~~ typical, add von Hippel article

Needs: GSS data

Figure 1.2: First Quadrant: General Information

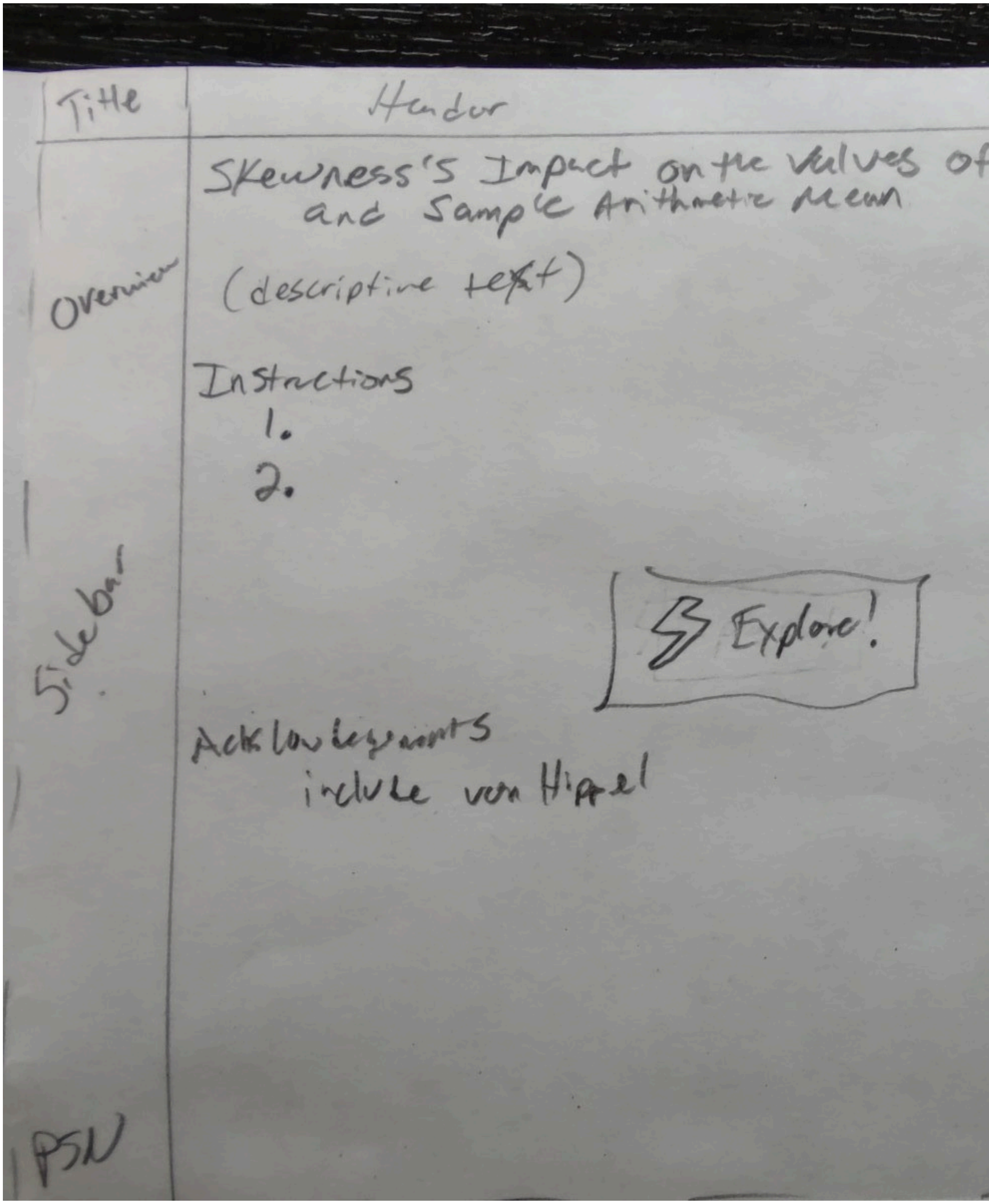


Figure 1.3: Second Quadrant: Overview Tab

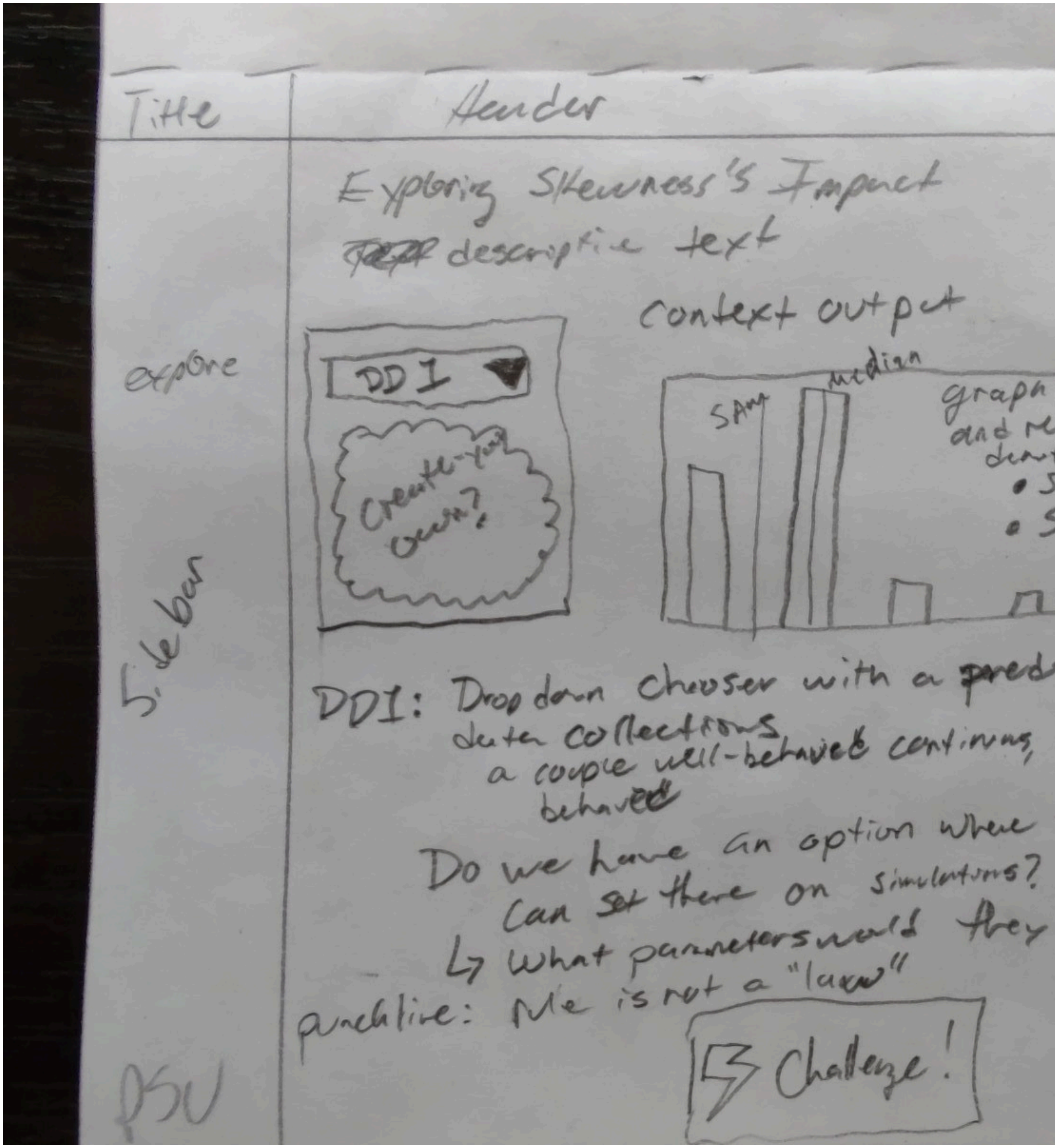

Title
Header
Exploring Skewness's Impact
descriptive text
context output
explore
DD I
Create-your own?
SAM
median
graph and re
dem
• S
• S
Sidebar
DDI: Drop down chooser with a ~~pred~~
data collections
a couple well-behaved continuous,
behaved
Do we have an option where
can set there on simulations?
↳ What parameters would they
punchline: Me is not a "law"
Challenge!
PSU

Figure 1.4: Third Quadrant: Exploration Tab

Social Survey, has a variety of attributes that we could use, include several that demonstrate what I want. For example, the number of adults (18+ y.o.) living in a household is positively skewed but the value of the *sample median* is larger than than the value of the *sample arithmetic mean*. Which each choice of the user makes, the histogram on the right will update to show the appropriate data, plus reference lines for the values of the two statistics. I might want to add a numerical display as well. I also want there to be some context helping the user to understand each choice.

I've left as an open question whether to enable a user to import/create their own data set on this tab. I'm currently leaning towards not allowing this. Additionally, I've marked something for a punchline but this will need more thought.

The fourth and final quadrant (Figure **??**) is an idea I'm still kicking around. This would depend on the specific audience and course goals for which users would go here. My idea is to extend the goal for the app to allow students to construct an understanding for what conditions might lead to $Sample\ Median\left(\mathcal{X}\right) < SAM\left(\mathcal{X}\right)$, $Sample\ Median\left(\mathcal{X}\right) = SAM\left(\mathcal{X}\right)$, and $Sample\ Median\left(\mathcal{X}\right) > SAM\left(\mathcal{X}\right)$.

At this moment, I'm leaning towards marking the Challenge Tab as being a future improvement/addition rather than being part of the initial version of the app.

1.3.2 Share Your Plan

Once you have a sketch, you'll want to share this plan with others to get ideas for suggestions, modifications, etc. Different people might be able to point you to code snippets you can reuse from other apps. More importantly, they might be able to help you think about things that you glossed over/missed and/or thought about but didn't articulate. By sharing these plans, we'll make better apps.

This is also the point in time in which the faculty will say whether you can go ahead and move to Step 4.

1.4 Step 4: Create Repository

Here you'll want to reference Section **??** on GitHub Repo Naming. Most app [full] titles don't make good Repo Names. This is where the potential short title comes into play. You'll need to make sure that there isn't already a repository that has that name. When you get a name figured out for the repository, you can go ahead and create the repo.

Make sure to set Git Ignore to `R`.

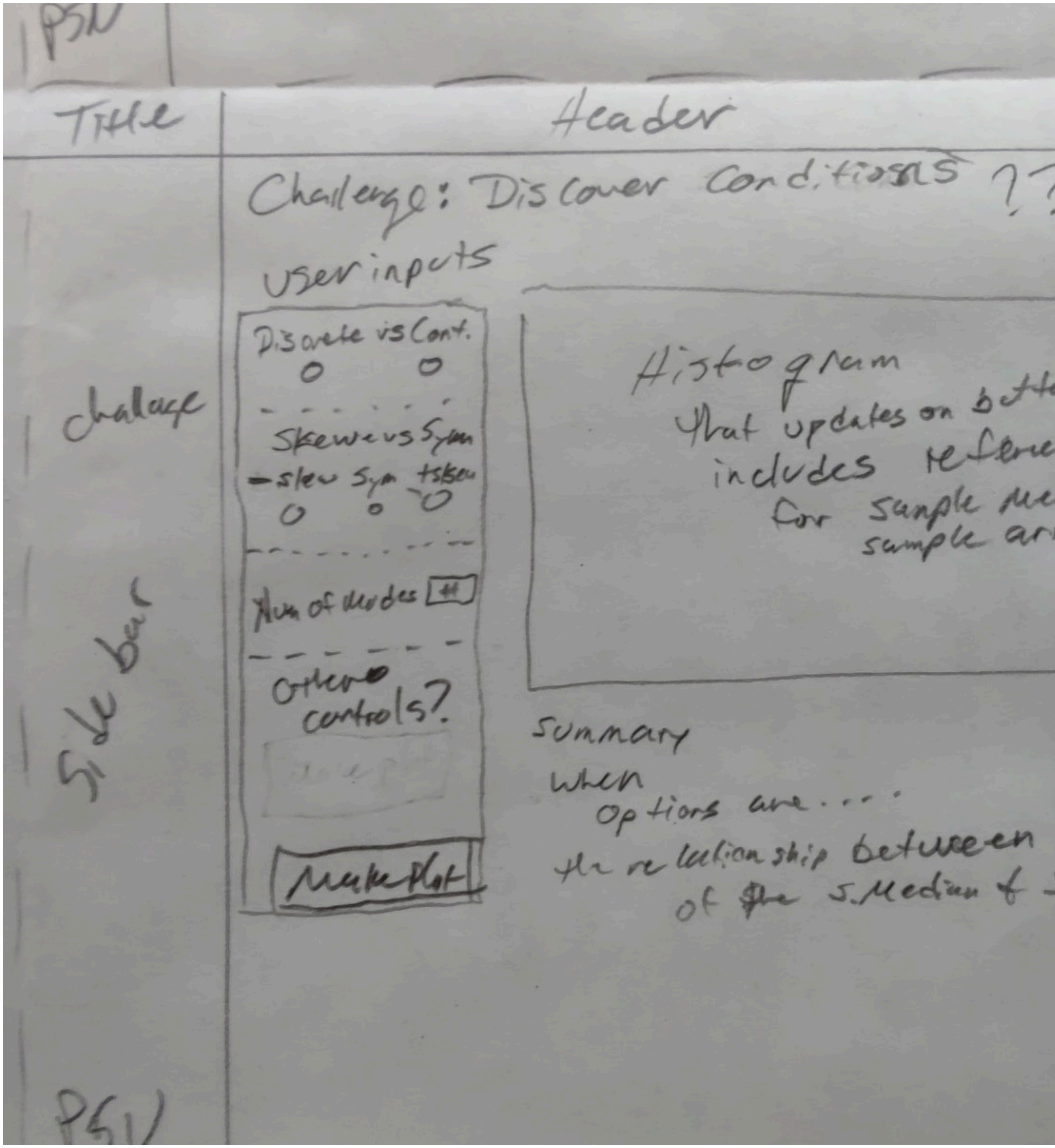


Figure 1.5: Fourth Quadrant: Challenge Tab

1.5 Steps 5-8: Write, Edit, Test, Debug

We recommend starting with a copy of the Sample App as the basis. This will give you a good skeleton to use for your new app.

(*Still under construction*)

1.6 Step 9: Larger Scale Testing

(*Still under construction*)

1.7 Steps 10-11: Pull Requestion and Tweaks

(*Still under construction*)

Chapter 2

Workflow

As you work on apps, both updating existing and developing new ones, moving through a workflow can help you organize yourself.

2.1 Revising Existing Apps

1. Read the Style Guide
2. Explore the existing apps in the book
3. When you identify an app you wish to work on,
    a. Go to that app's repository in GitHub
    b. Create new issues on GitHub to log both bugs and suggestions for improvements
    c. Create an issue specifically if the App currently does not abide by this Style Guide
    d. Optional–assign yourself to the issue
4. Download the current version of the app by using either RStudio or GitHub Desktop
5. Create a new branch for your developments
6. Begin editing the code
7. As you edit, be sure to reference this Style Guide and test your code locally via Run App in RStudio
8. Periodically push your edits to your development branch; don't forget to add commit messages and reference any issues
9. When you're ready to do larger scale testing you'll need to publish your App to the TLT RStudio Connect server (see Section **??**)
10. When you've reached a point where you've finished editing, push your most recent commit to your development branch and then create a Pull Request; assign Neil as a reviewer
11. Make any requested changes and re-push to your development branch; update the Pull Request with a new comment

If everything checks out, then we'll merge your development branch with the master branch and schedule formal deployment of your App.

2.2 Creating New Apps

To see this workflow in action, check out Chapter **??**.

1. Read the Style Guide.
2. Identify a topic for your new App.
3. Sketch out your plans for the App. **This should occur BEFORE you start coding.** Be sure that you include:
   a. Suggested Title
   b. Goal of the app
   c. What will the user be doing (i.e., potential inputs)
   d. What will the user be experiencing (i.e., potential outputs)
   e. Relationships between various elements.
4. When your plan **gets approval**, create a new repository on GitHub (see Section **??**); you can initially load the Sample App template to this repository, if you wish.
5. Begin writing the code.
6. As you edit, be sure to reference this Style Guide and test your code locally via Run App in RStudio.
7. When you get the basic structure of your App set up, push your code to GitHub and create a new development branch for your continued editing. You can open new issues on your app as you go.
8. Periodically push your edits to your development branch. Don't forget to add commit messages and reference any issues.
9. When you're ready to do larger scale testing you'll need to publish your App to the TLT RStudio Connect server (see Section **??**).
10. When you've reached a point where you've finished editing, push your most recent commit to your development branch and then create a Pull Request. Assign Neil as a reviewer.
11. Make any requested changes and re-push to your development branch. Update the Pull Request with a new comment.

If everything checks out, then we'll merge your development branch with the master branch and schedule formal deployment of your App.

2.2.1 GitHub Repo Names

Each Shiny App has its own repository (repo) on GitHub. As you begin to create new apps, you'll have to create a new repo on GitHub for each one. The name of that repo is extremely important as this will play a critical role in establishing the URL for your App. To this end, you need to adhere to following guidelines:

- Use Title Case (not camelCase)
- Use underscores (_) instead of spaces

- Match the App name as closely as possible
- You have a 100 character limit

While we can change repo names, doing so results in a large number of edits that have to be made. Thus, think carefully about how you are going to name your repository.

If you come across a repo that is poorly named, please make an issue and provide some suggestions for new names.

2.3 Testing Your App

To test your App beyond your local machine, we will be making use of TLT's RStudio Connect platform. You will need to follow these directions.

2.3.1 Set Up/Login to the VPN

Please refer to the following PSU Knowledge Basis Articles:

- VPN for Mac OS
- VPN for Windows
- VPN for Apple iOS
- VPN for Android

You MUST be logged into the PSU VPN to both upload and test your App on each device.

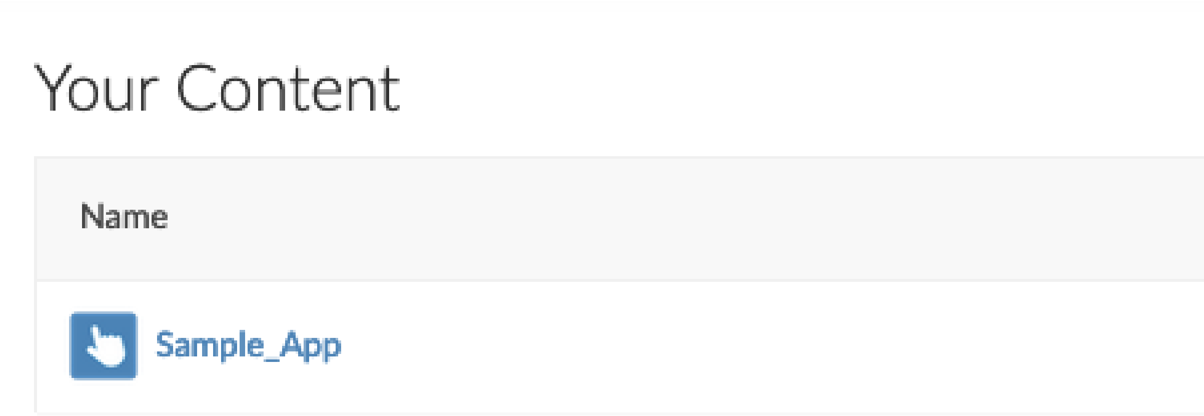
2.3.2 Connecting Your RStudio to TLT's RStudio Connect

You only need to do this step once.

1. After you've logged into the PSU VPN, go to TLT's RStudio Connect and log in using your PSU ID.
2. Once logged in, click on the Publish button as shown in the green box of Figure **??**
3. Select Shiny App and a pop up window will appear.
4. Follow the steps in this window (especially Step 4) to set up the connection.

2.3.2.1 Checking You've Connected

You can check to see if you've connected by going into RStudio's Options and looking at the Publishing tab. You should see a similar entry as what is in the green box of Figure **??**:



Your Content

Name

Sample_App

Figure 2.1: Click the Publish Button

2.3.3 Publishing Your App for Testing

Make sure that you are connected to PSU's VPN and that you've already connected your RStudio to TLT's server.

Click on the Publishing Icon, located just to the right of the Run App button. Be sure to select the TLT Server.

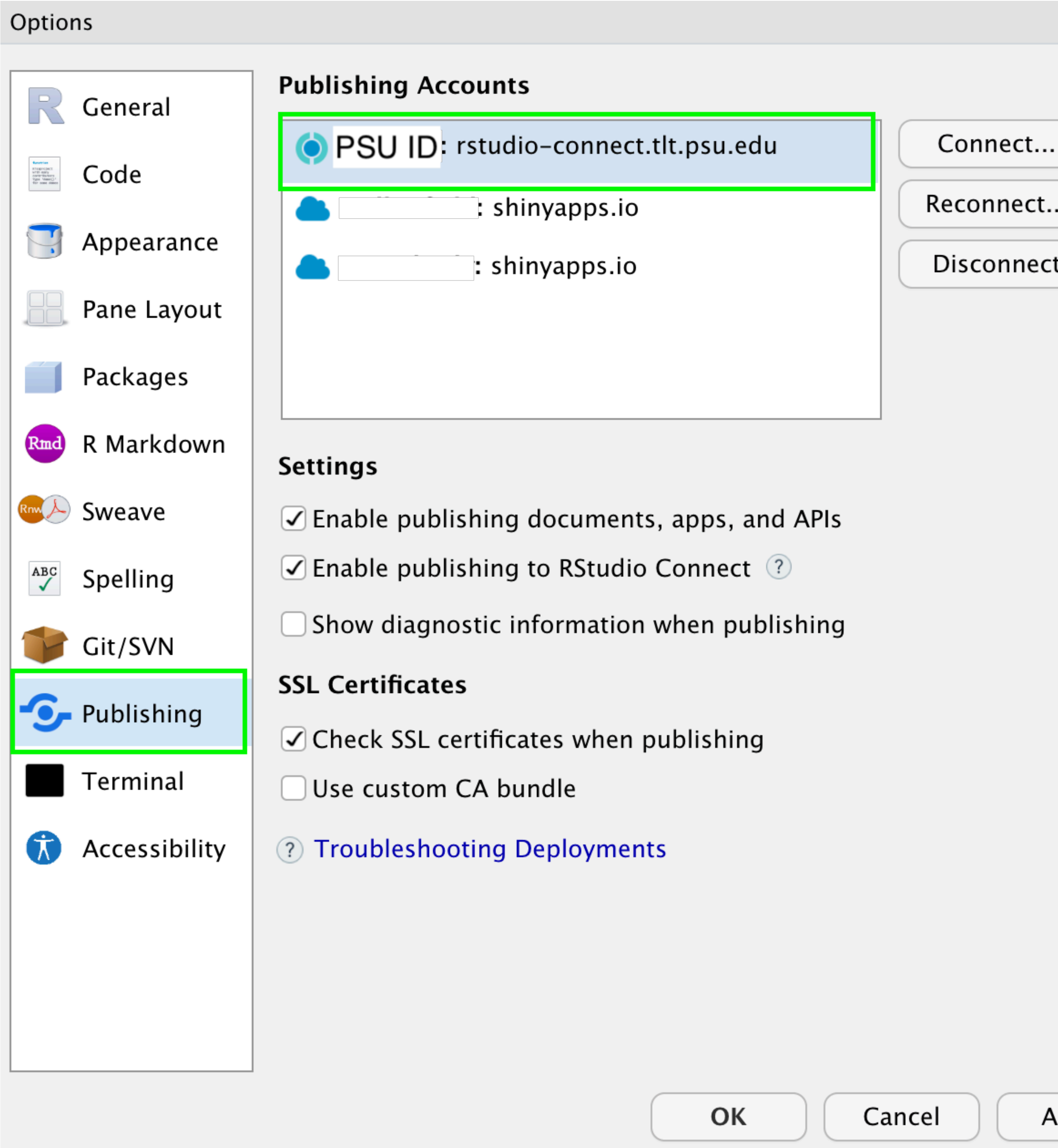
2.3.4 Configuring for Testing

Once your App has been published, a new window should open in your browser that shows your App plus the optional controls.

In Figure **??**, you will need to change a setting to enable others (people and devices) to test your app. Click on the Access tab and set Who can view this application to Anyone-no login required. (These are highlighted in the green boxes.) Keep in mind that all users/devices MUST first connect to the PSU VPN to access your app.

If you want, you can add collaborators as I did in the example (orange box). This is not required.

The most important piece is the Content URL (marked with the blue star). You'll need to copy this URL and give that URL out to your testers. This will allow them easily access your app, regardless of the type of device they are using.



Publishing Accounts

PSU ID: rstudio-connect.tlt.psu.edu

[redacted]: shinyapps.io

[redacted]: shinyapps.io

Settings

☑ Enable publishing documents, apps, and APIs

☑ Enable publishing to RStudio Connect ?

□ Show diagnostic information when publishing

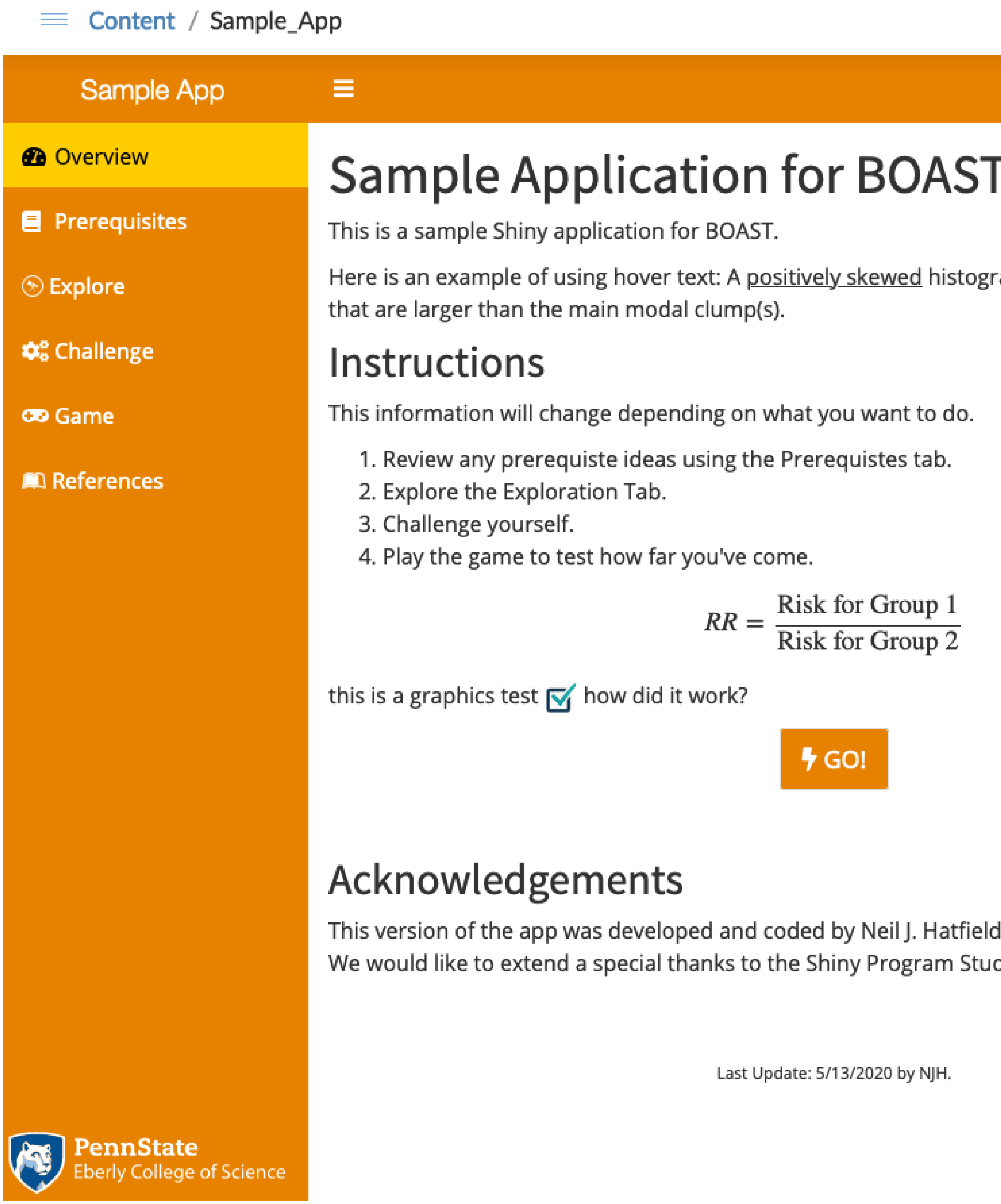
SSL Certificates

☑ Check SSL certificates when publishing

□ Use custom CA bundle

? Troubleshooting Deployments

Figure 2.2: Check Your Publishing Profile



Sample Application for BOAST

This is a sample Shiny application for BOAST.

Here is an example of using hover text: A positively skewed histogra
that are larger than the main modal clump(s).

Instructions

This information will change depending on what you want to do.

1. Review any prerequiste ideas using the Prerequistes tab.
2. Explore the Exploration Tab.
3. Challenge yourself.
4. Play the game to test how far you've come.

$$RR = \frac{\text{Risk for Group 1}}{\text{Risk for Group 2}}$$

this is a graphics test ☑ how did it work?

Acknowledgements

This version of the app was developed and coded by Neil J. Hatfield
We would like to extend a special thanks to the Shiny Program Stud

Last Update: 5/13/2020 by NJH.

Figure 2.3: Successful Publish and Settings

2.3.5 Check the Logs

As you test your App, you'll want to look for any error messages and/or warnings that get generated. Click on the Logs tab (red box in Figure **??**) to view.

2.3.6 Problems?

If you run into problem either publishing your App or getting the App to launch on the TLT, please reach out to Neil and Bob.

Part II

Getting Ready to Code

Chapter 3

The Basics

Before you get too far into the Style Guide, we would like for you to take a moment and ensure that you have the following tools at your disposal.

3.1 Getting Started Checklist

1. Ensure that you have all of necessary accounts and if not, put in a request to Bob (rpc5102).

   a. DataCamp

   b. GitHub

   c. EducationShinyAppTeam
   d. BOAST in Microsoft Teams (tied to your PSU AccessID)

2. Ensure that you have all of the required software

   a. R (version 3.5.* minimum, version 4.0.0 preferred)
   b. RStudio Desktop (most current version preferred)

3. Additional Software that we strongly recommend

   a. GitHub Desktop

4. R Packages–here are some basic packages that everyone will need; be sure to install their dependencies

   ```
   install.packages(c("devtools", "DT", "ggplot2", "shiny", "shinyBS",
                      "shinyWidgets", "tidyverse", "tippy"))
   ```

5. A copy of the Sample App (see below)

3.2 boastUtils Package

Bob created the boastUtils package to automate much of the design and development process. This will not only reduce the amount of work you'll need to do, it'll also make apps more consistent.

Starting Summer 2020, you will be required to you make use of this tool.

Please check out the package's GitHub page for instructions on installing and usage.

```r
library(devtools)
devtools::install_github("EducationShinyAppTeam/boastUtils")
```

3.3 Sample App

Bob and Neil have created a Sample App repository that you can use as a template for your own apps. To get started, clone the Sample_App template repo found on GitHub. This will provide you with a skeleton for organizing your files as well as your code.

There are several methods you can use:

- Direct Download (Basic)
- GitHub Desktop (Recommended)
- Command Line (Advanced)
- RStudio Project

3.3.1 Direct Download

You can download this repository directly by visiting:

> https://github.com/EducationShinyAppTeam/Sample_App/archi
> ve/master.zip

3.3.2 GitHub Desktop

If you are using GitHub Desktop and have linked your account that has access to the EducationShinyAppTeam repository, you can do the following from inside GitHub Desktop:

1. Bring up the Clone Repository Menu (File -> Clone Repository…)

2. Enter Sample_App in the search bar and select the option that says Sample_App (not sampleapp)
3. Click the Choose… button for the local path (this is where you want to the clone to live on your computer)
4. Click the Clone button.

See also:

- GitHub Desktop Help

3.3.3 Command Line

Enter the following command in your terminal:

```
git clone git@github.com:EducationShinyAppTeam/Sample_App.git
```

See also:

- Git Handbook
- Resources to learn Git
- Happy Git and GitHub for the userR

3.3.4 RStudio Project

You can also use RStudio to connect to and copy repositories from GitHub. There are a few extra steps that you'll need to go through compared to other methods. We recommend that you use GitHub Desktop.

1. In your browser, navigate to the repository you want to copy; in this case the Sample_App repo.
2. Launch RStudio and create a new project.
3. Select the Version Control option and then Git.
4. Switch back to your browser and click on the Clone or Download button. There should be a line of text starting with `git@github` that you can copy. Do so.
5. Returning to RStudio, paste the get you just copied into the Repository URL field.
6. Adjust the directory name and the subdirectory (where this project will live on your machine) as you see fit.
7. Click the Create Project button.

This method will ensure that you will be able to successfully push and pull changes to GitHub. For additional help, Neil has created a video "Using RStudio to Connect to GitHub" which walks you through the process.

Chapter 4

The Exploring the BOAST Template

Now that you've gotten your computer set up and you've made yourself a workflow, you're nearly ready to begin working on apps for BOAST. This Style Guide is the primary key to getting start but the BOAST Template is the secondary key to your success.

4.1 Structure of the BOAST Template Repository

To get started, you'll want to take a look at the BOAST App Template Repository either online or clone the repository ("repo") to your computer. There are several key items in the repo that you'll need to be aware of:

- `app.R`: this is where you will write the code for your app.
  - Note: Older apps (i.e., non-compliant with this Style Guide) might have `ui.R` and `server.R` files instead. You'll need to convert them if you are doing an update.
- `www` folder: this will be where you store image files and other non-data, non-R files for your app.
- `docs` folder: this is where several key files are stored for your including the License, a Readme, and a screenshot.
- `DESCRIPTION`: this file is where you'll need to place some key information about your app
- Additional files you may ignore: `.gitignore`, `.lintr`.

As you build your app (or work with existing apps), you'll add various files to the repo. For some of the older apps, part of the maintainer's job will be move