COURSE INTRO

& GITHUB FOR DATA SCIENCE

Shiny from RStudio

OUTLINE

‣ Course intro
  ‣ Instructor
  ‣ Teacher's Assistant
  ‣ Course overview

‣ GitHub for Data Science
  ‣ Read: Introduction to Github for Data Scientists by Rebecca Vickery
    ‣ https://towardsdatascience.com/introduction-to-github-for-data-scientists-2cf8b9b25fba
‣ R Projects

Course Intro

HELLO

my name is

GEOFFREY ARNOLD

geoffreylarnold @twitter

gla@andrew.cmu.edu

ABOUT ME

‣ Senior Digital Services Analyst

  ‣ City of Pittsburgh

‣ MSPPM 2015

  ‣ Heinz College

HELLO
my name is

MALVIKA
SINGH

malvikas@andrew.cmu.edu

CLASSES & OFFICE HRS

- Classes

  - Tue/Thu 4:30-5:50

  - Location: HBH 1006

- Office Hours

  - Geoffrey Arnold

    - Thursdays: 6:15-7:00

    - Location: HBH Lounge

  - Malvika Singh

    - Tu 10:30am-12pm

    - Th 3pm-4:30

    - Location: HBH 1109

COURSE OUTLINE

Weeks 1, 2, & 3 - Shiny

Week 4 - Maps with Leaflet

Week 5 - Advanced Shiny

Week 6 - SQL and API's

Week 7 - Human Centered Design

Intro to Github

"Experience with version control is fast becoming a requirement for all data scientists"
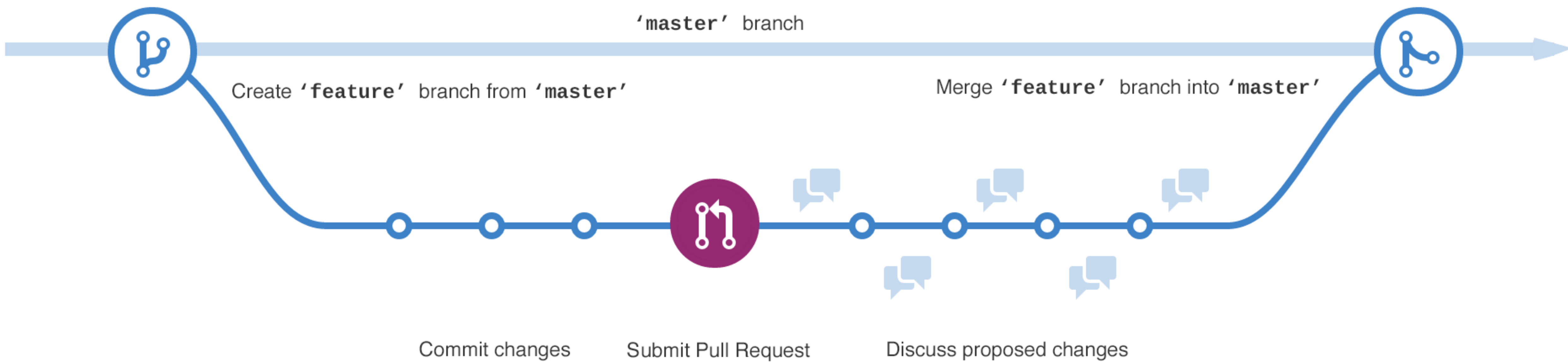–Rebecca Vickery

WHAT IS GITHUB?

- Git is a Version Control Software

- Github stores the files for your project in a remote location and checks the differences as you change your code

- This allows you to roll back to previous versions of your project if you need to go back

- Makes sharing and collaboration much easier using the Github website

Shiny from R Studio™

OTHER VERSION CONTROL

There are other kinds of version control software, GitLab also uses git.

We will be using Github in this course, but if in a future life, you might use these others.

Github Desktop
and the Web

‣ Download Github Desktop: https://desktop.github.com/

‣ Sign up for Github: https://github.com/join

‣ Got to course page: https://github.com/orgs/rforoperations2019/

  ‣ Clone Class 2 repo: https://github.com/rforoperations2019/Class-02-Shiny-Intro

  ‣ Create your own branch as your CMU username

5ₘ 00ₛ

Github and Projects in RStudio

"R Projects are great."

–Geoffrey Arnold

R PROJECTS

- So what is all this?

  - Avoid messy environment

  - Keep custom functions in check

  - Don't lose your work just because you want to do something else

- Info: https://support.rstudio.com/hc/en-us/articles/200526207-Using-Projects

HOW DO PROJECTS WORK?

- R typically saves your environment information in a default location (typically your Documents folder)

- When you create a project it gets its own .RData file for the project in the Directory/folder you created

- This is also the default working directory for your project, so no need to put all of the folders its in

  - Simply load objects by name if they're in the project folder

‣ Create a "New Project"

   ‣ Select "New Directory"

      ‣ Select "New Project"

         ‣ Make sure the "Create a git repository" is selected

         ‣ Give the project any name you want

            ‣ Click "Create Project"

               ‣ Look at the "Git" tab
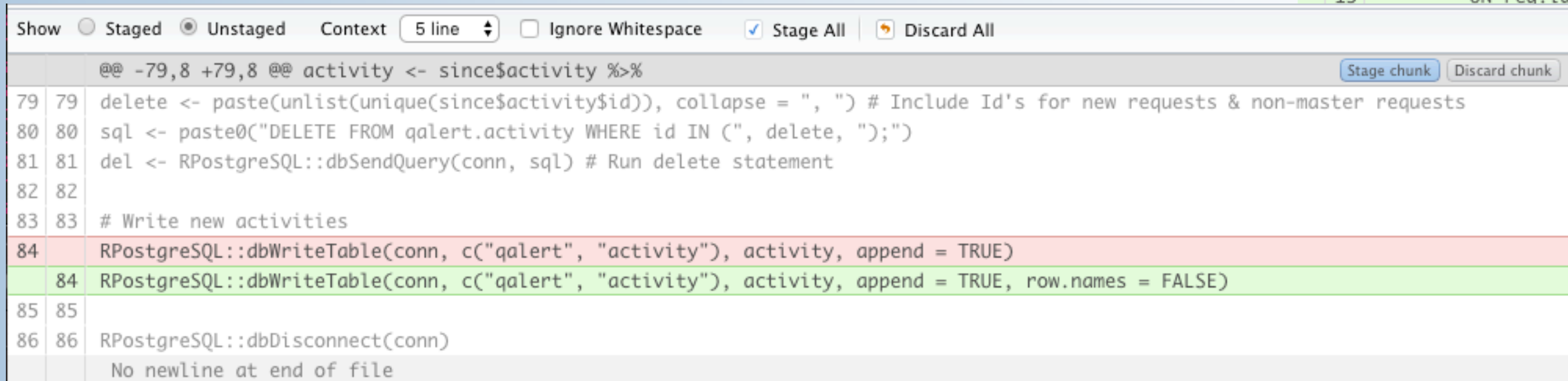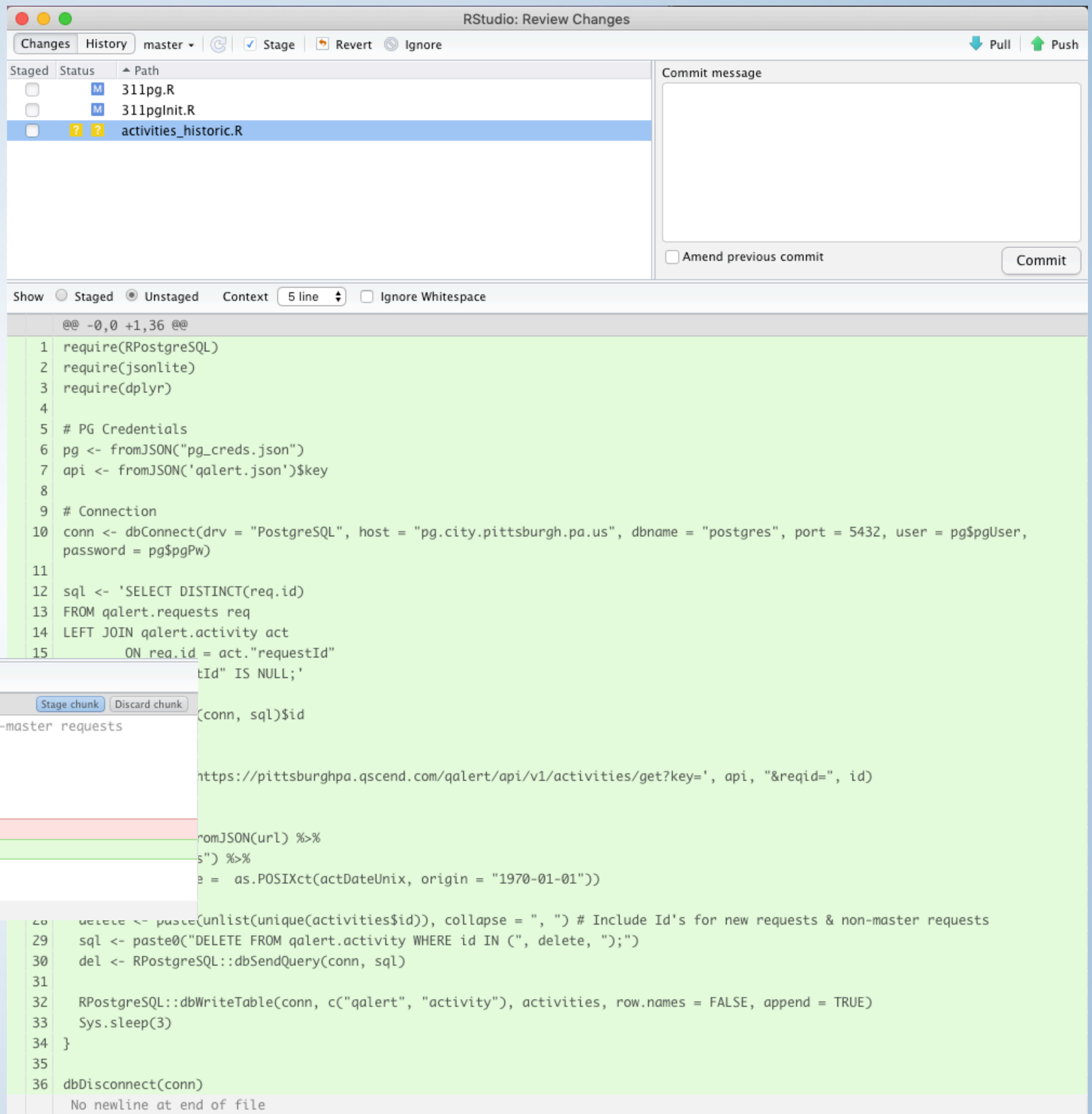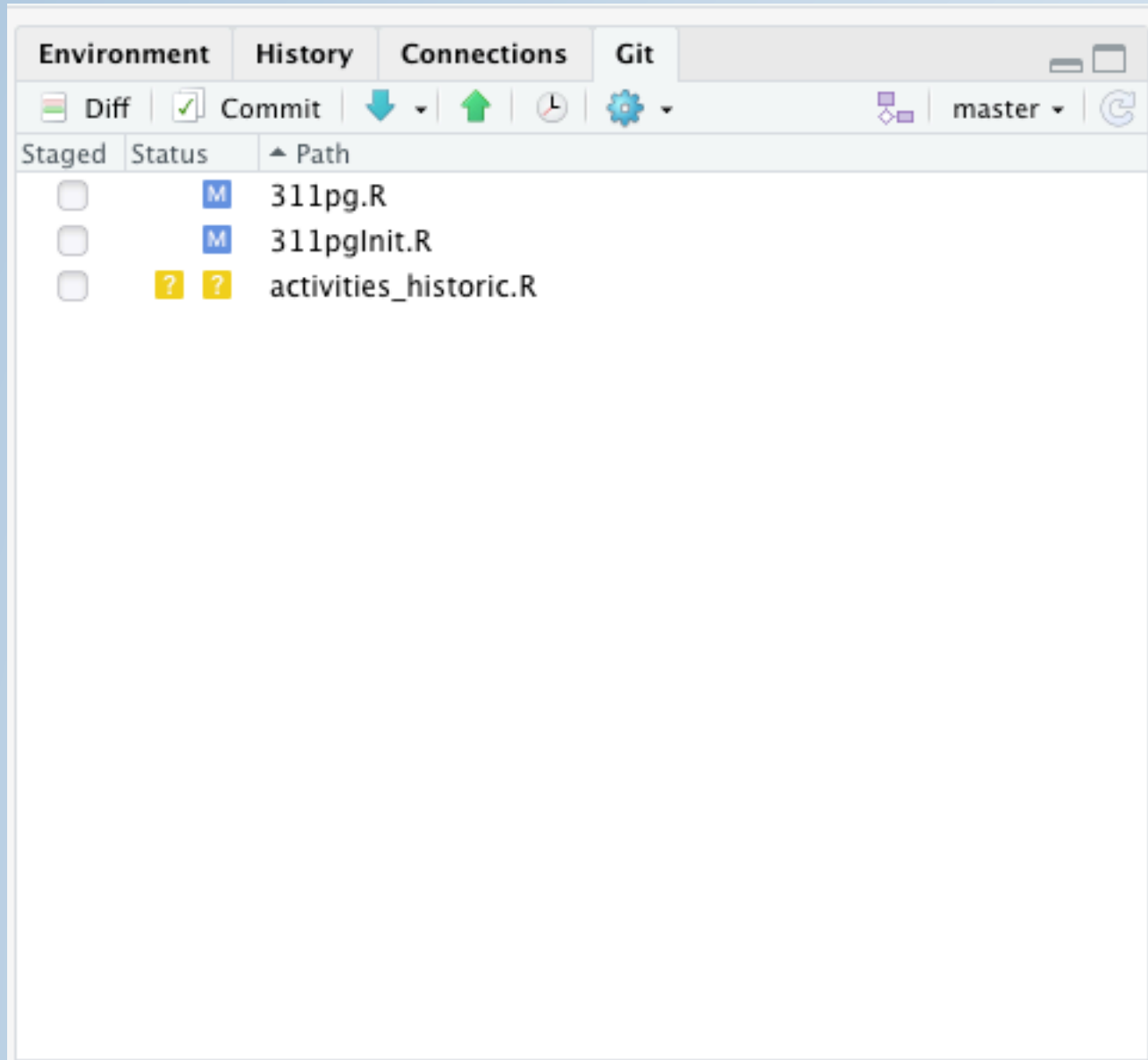
5$_m$ 00$_s$

‣ Click "File"

   ‣ Click "New File"

      ‣ Select "Shiny Web App"

         ‣ Give the application a name and click "Create"

      ‣ Click the "Git" tab

         ‣ What's change?

5_m 00_s

Environment   History   Connections   Git

Diff   ☑ Commit   ⬇ ▾   ⬆   🕐   ⚙ ▾   master ▾   🔄

Staged   Status   ▲ Path
☐   M   311pg.R
☐   M   311pgInit.R
☐   ⚠ ⚠   activities_historic.R

RStudio: Review Changes

Changes   History   master ▾   🔄   ☑ Stage   ↩ Revert   ⦸ Ignore   ⬇ Pull   ⬆ Push

Staged   Status   ▲ Path   Commit message
☐   M   311pg.R
☐   M   311pgInit.R
☐   ⚠ ⚠   activities_historic.R

☐ Amend previous commit   Commit

Show ○ Staged ● Unstaged   Context 5 line ⇕   ☐ Ignore Whitespace

@@ -0,0 +1,36 @@
```
 1  require(RPostgreSQL)
 2  require(jsonlite)
 3  require(dplyr)
 4
 5  # PG Credentials
 6  pg <- fromJSON("pg_creds.json")
 7  api <- fromJSON('qalert.json')$key
 8
 9  # Connection
10  conn <- dbConnect(drv = "PostgreSQL", host = "pg.city.pittsburgh.pa.us", dbname = "postgres", port = 5432, user = pg$pgUser,
    password = pg$pgPw)
11
12  sql <- 'SELECT DISTINCT(req.id)
13  FROM qalert.requests req
14  LEFT JOIN qalert.activity act
15          ON req.id = act."requestId"
                tId" IS NULL;'
                (conn, sql)$id
            https://pittsburghpa.qscend.com/qalert/api/v1/activities/get?key=', api, "&reqid=", id)
            romJSON(url) %>%
            s") %>%
            e =  as.POSIXct(actDateUnix, origin = "1970-01-01"))
28      delete <- paste(unlist(unique(activities$id)), collapse = ", ") # Include Id's for new requests & non-master requests
29      sql <- paste0("DELETE FROM qalert.activity WHERE id IN (", delete, ");")
30      del <- RPostgreSQL::dbSendQuery(conn, sql)
31
32      RPostgreSQL::dbWriteTable(conn, c("qalert", "activity"), activities, row.names = FALSE, append = TRUE)
33      Sys.sleep(3)
34  }
35
36  dbDisconnect(conn)
        No newline at end of file
```

Show ○ Staged ● Unstaged   Context 5 line ⇕   ☐ Ignore Whitespace   ☑ Stage All   ↩ Discard All

@@ -79,8 +79,8 @@ activity <- since$activity %>%   Stage chunk   Discard chunk
```
79  79  delete <- paste(unlist(unique(since$activity$id)), collapse = ", ") # Include Id's for new requests & non-master requests
80  80  sql <- paste0("DELETE FROM qalert.activity WHERE id IN (", delete, ");")
81  81  del <- RPostgreSQL::dbSendQuery(conn, sql) # Run delete statement
82  82
83  83  # Write new activities
84      RPostgreSQL::dbWriteTable(conn, c("qalert", "activity"), activity, append = TRUE)
    84  RPostgreSQL::dbWriteTable(conn, c("qalert", "activity"), activity, append = TRUE, row.names = FALSE)
85  85
86  86  RPostgreSQL::dbDisconnect(conn)
        No newline at end of file
```

‣ Go to "Global Options"

  ‣ Click "Git/SVN"

    ‣ Ensure "Enable version control interface for Studio projects" is selected

      ‣ Click "Create RSA Key…"

        ‣ Click "Create"

          ‣ Click "View public key" and copy key

‣ Go to https://github.com/settings/keys

  ‣ Click "New SSH key"

    ‣ Paste key in text box and give your key a name

      ‣ Click "Add SSH Key"

  ‣ If you have two factor authorization turned on for GitHub (people with previous GitHub accounts may have this turned on) you will need your Personal Access Token to login later

  ‣ Everyone else, your GitHub login and password will be important when logging in later.

5_m 00_s
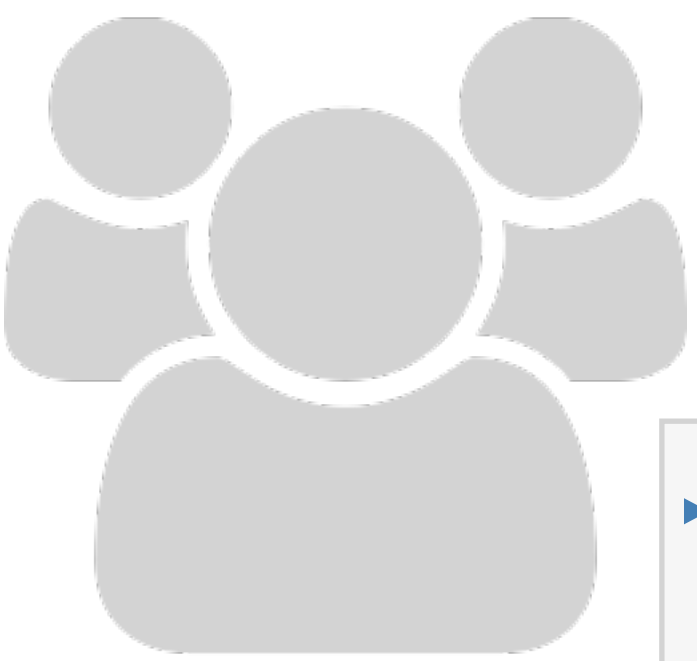
‣ Click the "Git" tab

  ‣ Click "Commit"

    ‣ Type a message into the "Commit message" box

      ‣ Stage the files by making sure they are selected on the left

        ‣ Click "Commit"

          ‣ Click "Push"

            ‣ Login to GitHub with either your password or personal access token

5m **00**s

Github in the command line / terminal

WHATS THE COMMAND LINE

‣ Command or terminal git commands are how git was first used.

‣ You don't need to know how to do these things as either the "Git" tab in RStudio or the Github desktop program provides a GUI (gooey user interface) for you.

‣ However, knowing the hard way to do something never hurt anybody.

‣ https://learngitbranching.js.org/

‣ Complete Introduction Sequences

   ‣ Git Commits

   ‣ Branching Git

   ‣ Merging in Git

5ᴍ 00ₛ