



SUSTAINERS OF THE
tidyverse



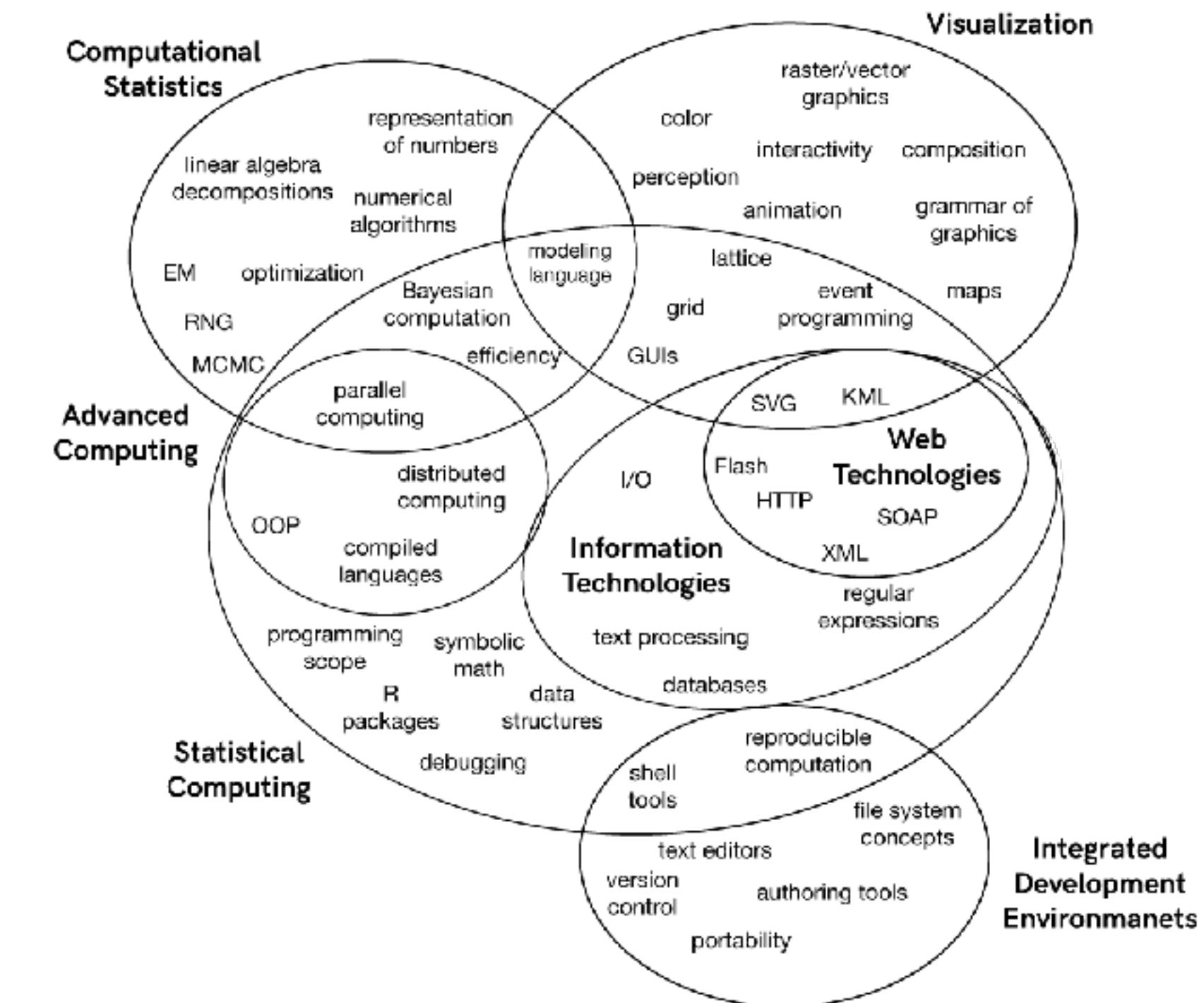
Mara Averick ([@dataandme](https://twitter.com/dataandme))

Tidyverse Developer Advocate, RStudio



Mara Averick

TIDYVERSE DEV ADVOCATE, RSTUDIO





Mara Averick

TIDYVERSE DEV ADVOCATE, RSTUDIO





SCIENCE & SOCIETY

Studying scientists...

“Like many social groups that do not reproduce themselves biologically, the experimental particle physics community renews itself by training novices.”

— Sharon Traweek, *Pilgrim's Progress: Male Tales Told During a Life in Physics*, 1988



OPEN SOURCE TOOLS

Create demand



Enterprise Products
Invest in



Strategy

58%

of company is
Engineering

60%

of engineering
is open source

```
>y<-25
```

```
>y R generation
```

```
[1] 25
```

The story of a statistical programming language that became a subcultural phenomenon. By **Nick Thieme**

25 years of R

- 1992 Robert Gentleman and Ross Ihaka (S)
- Statistical programming language
- Maintained by R Core
- 2000 R version 1.0.0
- Over 16,000 packages on CRAN

What is the tidyverse?

The tidyverse is an opinionated collection of R packages designed for data science.

All packages share an underlying design philosophy, grammar, and data structures.

Import

readr
readxl
haven
xml2

Tidy → Transform

tibble
tidyverse
dplyr
forcats
hms
lubridate
stringr

purrr
magrittr

Program

Visualise

ggplot2

recipes
rsample
tidybayesian
yardstick

Model

broom
modelr

shiny
rmarkdown

Communicate

Import

readr
readxl
haven
xml2

Tidy → Transform

tibble
tidyverse
purrr
magrittr
dplyr
forcats
hms
lubridate
stringr

Program

Visualise

ggplot2



recipes
rsample
tidybayesian
yardstick

Model

broom
modelr



shiny
rmarkdown



Communicate

Import



Tidy →

Store data
consistently

Transform

Create new variables & new summaries



Visualise

Surprises, but doesn't scale



Model

Scales, but doesn't (fundamentally) surprise

Communicate

Automate

Understand

TIDY TOOLS

Functions
should be...

SIMPLE

Do one thing and do it well.

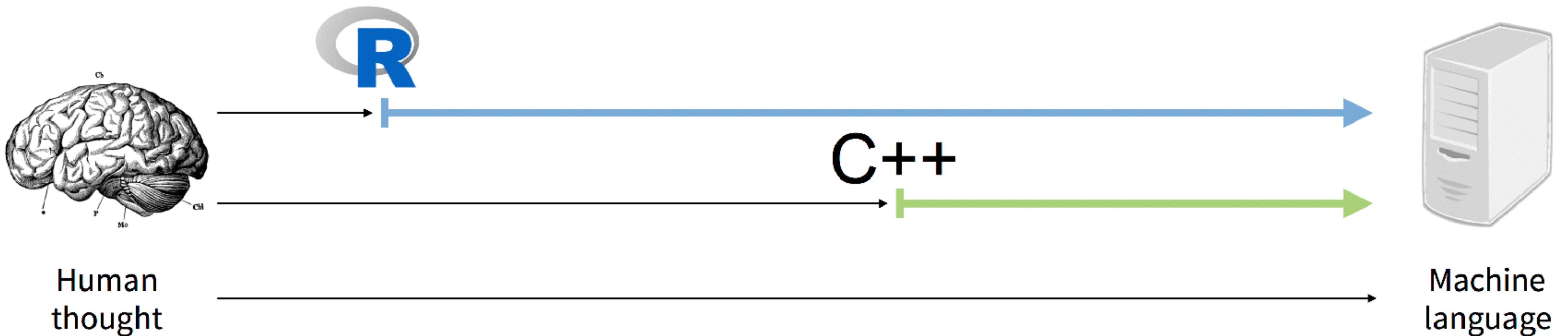
COMPOSABLE

Combine with other functions for multi-step operations.

DESIGNED FOR HUMANS

Use evocative verb names, making them easy to remember.

A computer language for scientists



Practitioner

Interactive
Easily detect & resolve problems

Programmer

Packaged
In production



Code is a conversation
Ambiguity can be tolerated

Code is a script
Fail early and often

Implicit

Explicit

**What does this mean for
contribution and sustainability?**

Key FOSS actors

GitHub HQ (SF) | June 19, 2017

- maintainers
- contributors
- consumers/users
- sustainers

Report team: Ben Nickolis, Pia Mancini, Justin Dorfman, Robert Gibb <<https://sustainoss.org/>>

Sustain

A ONE DAY CONVERSATION
for Open Source Software sustainers

THE REPORT

“When we talk about sustainability, we are talking both and equally about the sustainability of resources and the sustainability of its people.”

THE REPORT

Sustain: key recommendations

- ✓ Create sustainable communities
- ✓ Free the maintainer
- ✓ Raise the value of non-code contributions

Sustaining the tidyverse...

- virtual spaces
- technical tools
- social norms

Sustaining the tidyverse...

- virtual spaces
- technical tools
- social norms

spoiler alert:
they're interrelated

Contribute to the tidyverse - Ti X +

← → C https://www.tidyverse.org/contribute/ ⚡ ⌂ ⌂

Tidyverse Packages Articles Learn Help Contribute

Contribute to the tidyverse

The tidyverse would not be possible without the contributions of the R community. No matter your current skills, it's possible to contribute back to the tidyverse.

Answer questions

The easiest way to help out is to answer questions. You won't know the answer to everything, but that's ok! Even just the acknowledgement that someone cares enough to try can be tremendously encouraging.

Many people asking for help, don't know about reprexes. A little education, and some help crafting a [reprex](#) can go a long way. You might not answer the question, but you'll help someone answer it more easily.

If you're interested in answering questions, some good places to start are the [RStudio community site](#), or the tidyverse tags on [Twitter](#) and [Stack Overflow](#). Just remember that while you might have seen the problem a hundred times before, it's new to the person asking it. Be patient, polite, and empathic.

File issues

If you've found a bug, first create a minimal [reprex](#). Spend some time trying to make it as minimal as possible: the more time you spend doing this, the easier it will be for the tidyverse team to fix it. Then file it on the GitHub repo of the appropriate package.

To be as efficient as possible, development of tidyverse packages tends to be very bursty. Nothing happens for a long time, until a sufficient quantity of issues accumulates. Then there's a burst of intense activity as we focus our efforts. That makes development more efficient because it avoids expensive context switching between problems. This process makes a good reprex particularly important because it might be multiple months between your initial report and when we start working on it. If you can't reproduce the bug, we can't fix it!

Contribute documentation

If you're a bit more experienced with the tidyverse and are looking to improve your open source development skills, the next step up

Contents

- [Answer questions](#)
- [File issues](#)
- [Contribute documentation](#)
- [Contribute code](#)

Upcoming events

rstudio::conf 2019
Austin, TX
Jan 15-18

rstudio::conf 2019 covers all things RStudio, including workshops to teach you the tidyverse, and talks to show you the latest and greatest features.

tidyverse developer day
Austin, TX
Jan 19

Help the tidyverse team improve our code and documentation. First-time contributors are welcome.

A screenshot of a web browser window displaying the 'Contribute to the tidyverse' page from the tidyverse.org website. The page has a dark header with the 'Tidyverse' logo and navigation links for 'Packages', 'Articles', 'Learn', 'Help', and 'Contribute'. The main content area features a large heading 'Contribute to the tidyverse' and a sub-section about contributing back to the R community. On the right side, there are links for 'Contents', 'Answer questions', and 'File issues'. The URL in the address bar is https://www.tidyverse.org/contribute/.

Contribute to the tidyverse

The tidyverse would not be possible without the contributions of the R community. No matter your current skills, it's possible to contribute back to the tidyverse.

Contents

[Answer questions](#)

[File issues](#)

<https://www.tidyverse.org/contribute/>

The tidyverse would not be possible without the contributions of the R community. No matter your current skills, it's possible to contribute back to the tidyverse.

Important because it might be multiple months between your initial report and when we start working on it. If you can't reproduce the bug, we can't fix it!

Jan 19

Help the tidyverse team improve our code and documentation. First-time contributors are welcome.

Contribute documentation

If you're a bit more experienced with the tidyverse and are looking to improve your open source development skills, the next step up

Contribute to the tidyverse - Ti X +

← → C https://www.tidyverse.org/contribute/ ⚡ ⌂ ⌂

Tidyverse Packages Articles Learn Help Contribute

Contribute to the tidyverse

The tidyverse would not be possible without the contributions of the R community. No matter your current skills, it's possible to contribute back to the tidyverse.

Answer questions

The easiest way to help out is to answer questions. You won't know the answer to everything, but that's ok! Even just the acknowledgement that someone cares enough to try can be tremendously encouraging.

Many people asking for help, don't know about reprexes. A little education, and some help crafting a [reprex](#) can go a long way. You might not answer the question, but you'll help someone answer it more easily.

If you're interested in answering questions, some good places to start are the [RStudio community site](#), or the tidyverse tags on [Twitter](#) and [Stack Overflow](#). Just remember that while you might have seen the problem a hundred times before, it's new to the person asking it. Be patient, polite, and empathic.

File issues

If you've found a bug, first create a minimal [reprex](#). Spend some time trying to make it as minimal as possible: the more time you spend doing this, the easier it will be for the tidyverse team to fix it. Then file it on the GitHub repo of the appropriate package.

To be as efficient as possible, development of tidyverse packages tends to be very bursty. Nothing happens for a long time, until a sufficient quantity of issues accumulates. Then there's a burst of intense activity as we focus our efforts. That makes development more efficient because it avoids expensive context switching between problems. This process makes a good reprex particularly important because it might be multiple months between your initial report and when we start working on it. If you can't reproduce the bug, we can't fix it!

Contribute documentation

If you're a bit more experienced with the tidyverse and are looking to improve your open source development skills, the next step up

Contents

- [Answer questions](#)
- [File issues](#)
- [Contribute documentation](#)
- [Contribute code](#)

Upcoming events

rstudio::conf 2019
Austin, TX
Jan 15-18

rstudio::conf 2019 covers all things RStudio, including workshops to teach you the tidyverse, and talks to show you the latest and greatest features.

tidyverse developer day
Austin, TX
Jan 19

Help the tidyverse team improve our code and documentation. First-time contributors are welcome.

Contribute to the tidyverse - Ti X +

← → C https://www.tidyverse.org/contribute/ ⚡ ⌂ ⌂

Tidyverse Packages Articles Learn Help Contribute

Contribute to the tidyverse

The tidyverse would not be possible without the contributions of the R community. No matter your current skills, it's possible to contribute back to the tidyverse.

Answer questions

The easiest way to help out is to answer questions. You won't know the answer to everything, but that's ok! Even just the acknowledgement that someone cares enough to try can be tremendously encouraging.

Many people asking for help, don't know about reprexes. A little education, and some help crafting a [reprex](#) can go a long way. You might not answer the question, but you'll help someone answer it more easily.

If you're interested in answering questions, some good places to start are the [RStudio community site](#), or the tidyverse tags on [Twitter](#) and [Stack Overflow](#). Just remember that while you might have seen the problem a hundred times before, it's new to the person asking it. Be patient, polite, and empathic.

File issues

If you've found a bug, first create a minimal [reprex](#). Spend some time trying to make it as minimal as possible: the more time you spend doing this, the easier it will be for the tidyverse team to fix it. Then file it on the GitHub repo of the appropriate package.

To be as efficient as possible, development of tidyverse packages tends to be very bursty. Nothing happens for a long time, until a sufficient quantity of issues accumulates. Then there's a burst of intense activity as we focus our efforts. That makes development more efficient because it avoids expensive context switching between problems. This process makes a good reprex particularly important because it might be multiple months between your initial report and when we start working on it. If you can't reproduce the bug, we can't fix it!

Contribute documentation

If you're a bit more experienced with the tidyverse and are looking to improve your open source development skills, the next step up

Contents

- [Answer questions](#)
- [File issues](#)
- [Contribute documentation](#)
- [Contribute code](#)

Upcoming events

rstudio::conf 2019
Austin, TX
Jan 15-18

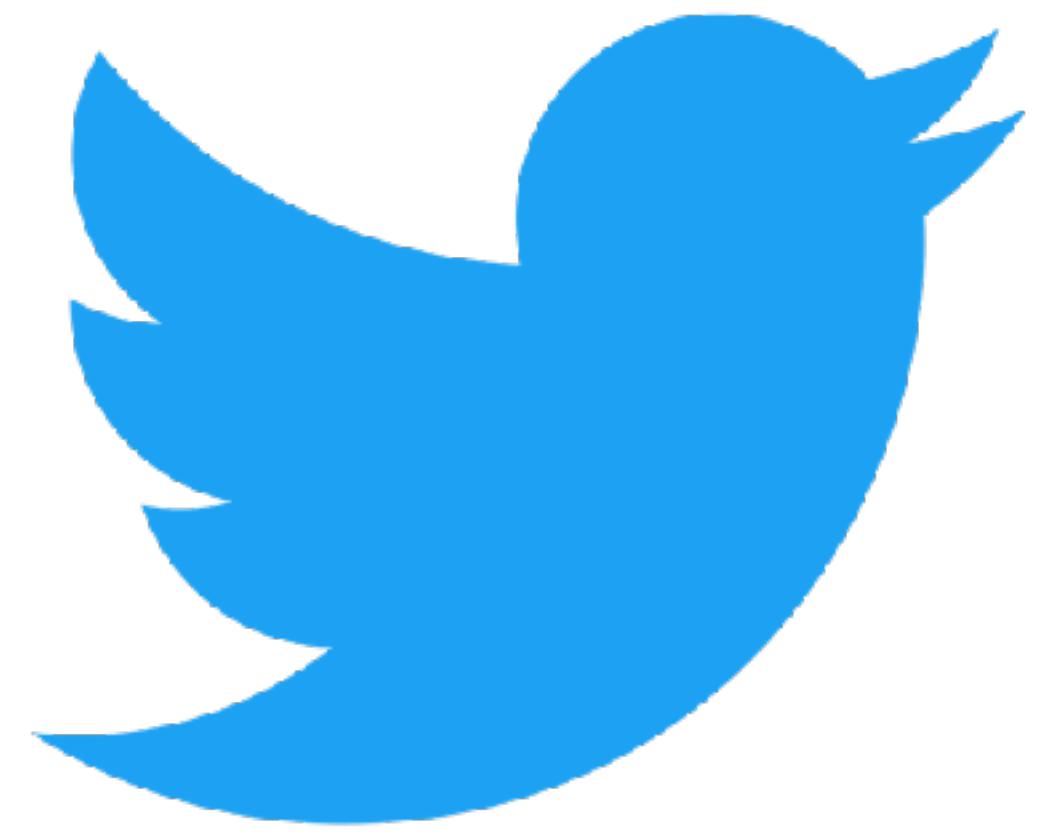
rstudio::conf 2019 covers all things RStudio, including workshops to teach you the tidyverse, and talks to show you the latest and greatest features.

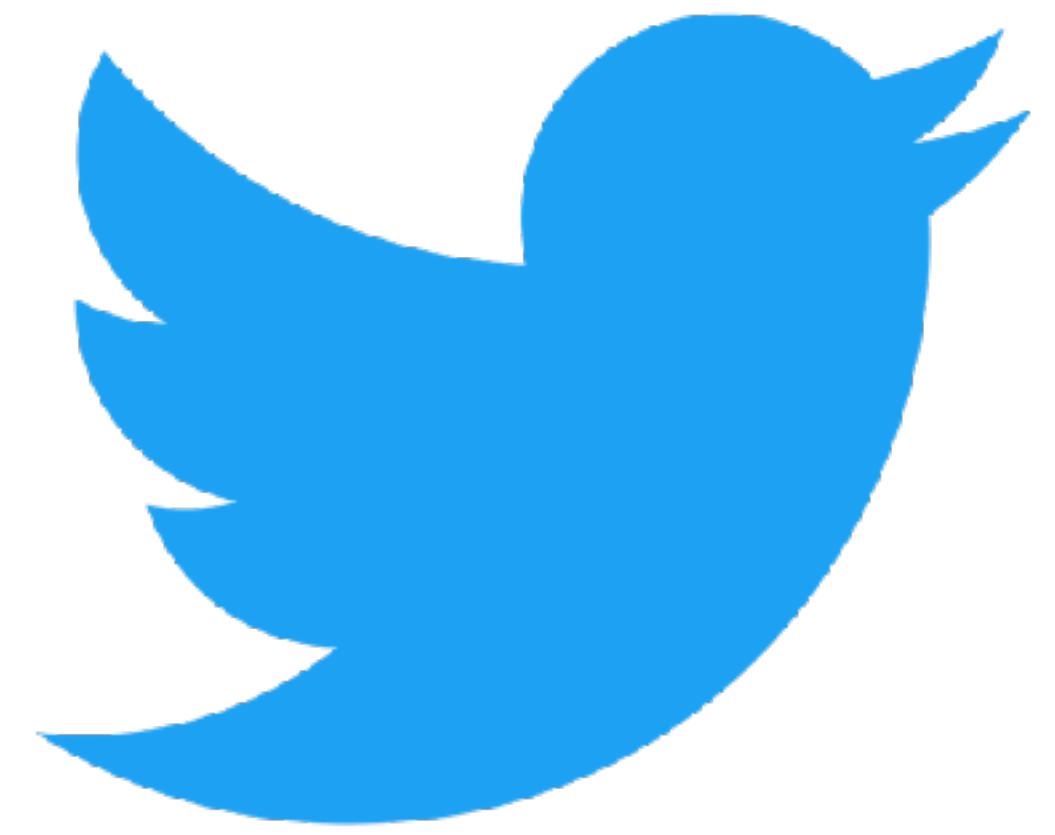
tidyverse developer day
Austin, TX
Jan 19

Help the tidyverse team improve our code and documentation. First-time contributors are welcome.

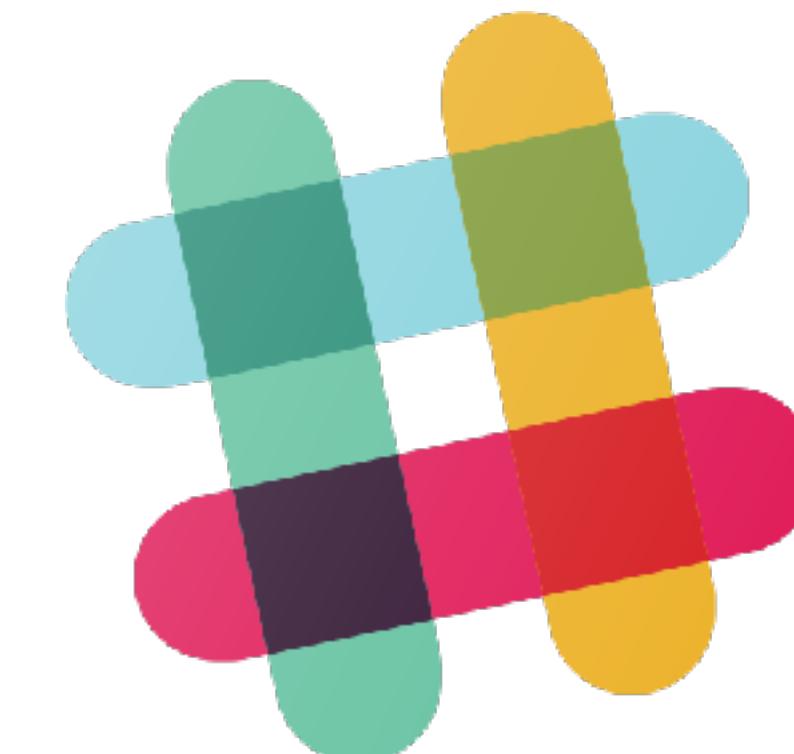
Where do people go to contribute?











and a bunch of other places...



RLadies



R4DS Online Learning Community



rOpenSci

Considerations

- Visibility

* Ford, D., Smith, J., Guo, P. J., & Parnin, C. (2016). Paradise unplugged: identifying barriers for female participation on stack overflow. 24th ACM SIGSOFT - FSE 2016, 846-857. <http://doi.org/10.1145/2950290.2950331>

Considerations

- Visibility
- Permanence

* Ford, D., Smith, J., Guo, P. J., & Parnin, C. (2016). Paradise unplugged: identifying barriers for female participation on stack overflow. 24th ACM SIGSOFT - FSE 2016, 846-857. <http://doi.org/10.1145/2950290.2950331>

Considerations

- Visibility
- Permanence
- Authority

* Ford, D., Smith, J., Guo, P. J., & Parnin, C. (2016). Paradise unplugged: identifying barriers for female participation on stack overflow. 24th ACM SIGSOFT - FSE 2016, 846-857. <http://doi.org/10.1145/2950290.2950331>

Considerations

- Visibility
- Permanence
- Authority
- Speed

* Ford, D., Smith, J., Guo, P. J., & Parnin, C. (2016). Paradise unplugged: identifying barriers for female participation on stack overflow. 24th ACM SIGSOFT - FSE 2016, 846-857. <http://doi.org/10.1145/2950290.2950331>

Considerations

- Visibility
- Permanence
- Authority
- Speed
- Peer parity (Ford et al. 2016)

* Ford, D., Smith, J., Guo, P. J., & Parnin, C. (2016). Paradise unplugged: identifying barriers for female participation on stack overflow. 24th ACM SIGSOFT - FSE 2016, 846-857. <http://doi.org/10.1145/2950290.2950331>

Considerations

- Visibility
- Permanence
- Authority
- Speed
- Peer parity*
- Prior social links (Casalnuovo et al. 2015)

* Casalnuovo, C., Vasilescu, B., Devanbu, P., & Filkov, V. (2015). Developer onboarding in GitHub: the role of prior social links and language experience. Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering - ESEC/FSE 2015, 817-828. <http://doi.org/10.1145/2786805.2786854>

"We don't do that here"

- Question phrasing
- Formatting posts
- Community triage
- Question framing
- Community culture of asking

"We Don't Do That Here": How Collaborative Editing with Mentors Improves Engagement in Social Q&A Communities

Denae Ford¹, Kristina Lustig², Jeremy Banks², Chris Parnin¹

¹North Carolina State University, Raleigh, NC, USA

²Stack Exchange, Inc., New York, NY, USA

{dford3, cjparnin}@ncsu.edu, klustig@stackoverflow.com, _@jeremy.ca

ABSTRACT

Online question-and-answer (Q&A) communities like Stack Overflow have norms that are not obvious to novice users. Novices create and post programming questions without feedback, and the community enforces site norms through public downvoting and commenting. This can leave novices discouraged from further participation. We deployed a month-long, just-in-time mentorship program to Stack Overflow in which we redirected novices in the process of asking a question to an on-site *Help Room*. There, novices received feedback on their question drafts from experienced Stack Overflow mentors. We present examples and discussion of various question improvements including: question context, code formatting, and wording that adheres to on-site cultural norms. We find that mentored questions are substantially improved over non-mentored questions, with average scores increasing by 50%. We provide design implications that challenge how socio-technical communities onboard novices across domains.

ACM Classification Keywords

H.5.3. Group and Organization Interfaces: Computer-supported cooperative work

Author Keywords

social Q&A, collaborative editing, e-mentoring, programming

INTRODUCTION

Building and maintaining active online communities is a difficult and well-documented problem across many community types [4, 15, 23]. For prospective community members, barriers such as learning community norms [24], overcoming technical hurdles [31], and resolving conflict [12] can be harmful to participation. In addition, these barriers may significantly affect people in marginalized groups, such as women and people of color, from fully participating in online communities [14]. This is especially pertinent for online programming communities.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyright for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

CHI 2018, April 21–26, 2018, Montreal, QC, Canada

© 2018 ACM. ISBN 978-1-4503-5620-5/18/04...\$15.00

DOI: <https://doi.org/10.1145/3173574.3174182>

Stack Overflow is the largest online programming community [25]. Each month, over 40 million people visit Stack Overflow, a social Q&A site, to learn about, ask, or answer over 14 million programming questions. Despite great popularity, there is evidence that negative behaviors and malfunctioning community mechanics can have long-term effects on site participation. For example, many questions go unanswered [30], and 90% of accepted answers provided by new users are self-answers. For Stack Overflow, "hostile" criticism and conflict [14, 13] is especially problematic for prospective members. As a result, a user may decide not to ask or answer a question for *fear of negative feedback* [14]. These problems can dissuade novices [31] and women [32] from participating in the community. On the other hand, active community members are interested in preserving community norms: not allowing duplicate questions, off-topic or non-closed questions, or poor quality answers. Community members need a mechanism for helping new users ask better questions, while reducing the hostility and negativity of otherwise well-meaning feedback.

In this paper, we applied theory related to learning and communities of practice to a social Q&A site, by using methods related to mutual engagement and formative feedback to improve novices' questions. Building on design claims for increasing engagement in online communities [19], we created *Help Rooms* with collaborative question drafts to enable novices to receive timely and formative feedback from mentors before posting their questions. Our *Help Rooms* work as follows: when a novice is about to post a question, they are asked if they want additional feedback from a mentor. If the novice responds positively, they are redirected to a room with a mentor who can help them edit their question. The mentor offers advice on how to phrase and ask their question so that it can be well received by the Stack Overflow community.

In a one-month online study, we implemented our mechanism for mentored question asking on Stack Overflow, and enabled 271 novices to receive help with their questions. As a result, we found that mentored questions were substantially improved over non-mentored questions. Average scores increased by 50%, resulting in fewer off-topic, deleted, and poor questions. Overall, for mentored questions, there was an increase in the amount of *good* questions asked, and reduction of *bad* questions asked by Stack Overflow standards. Novices surveyed agreed that they feel more comfortable posting on Stack Overflow after their participation (median = 4 on a 5-point Likert

"We don't do that here"

- Question phrasing
- Formatting posts
- Community triage
- Question framing
- Community culture of asking

format code as code

"We Don't Do That Here": How Collaborative Editing with Mentors Improves Engagement in Social Q&A Communities

Denae Ford¹, Kristina Lustig², Jeremy Banks², Chris Parnin¹

¹North Carolina State University, Raleigh, NC, USA

²Stack Exchange, Inc., New York, NY, USA

{dford3, cjparnin}@ncsu.edu, klustig@stackoverflow.com, _@jeremy.ca

ABSTRACT

Online question-and-answer (Q&A) communities like Stack Overflow have norms that are not obvious to novice users. Novices create and post programming questions without feedback, and the community enforces site norms through public downvoting and commenting. This can leave novices discouraged from further participation. We deployed a month-long, just-in-time mentorship program to Stack Overflow in which we redirected novices in the process of asking a question to an on-site *Help Room*. There, novices received feedback on their question drafts from experienced Stack Overflow mentors. We present examples and discussion of various question improvements including: question context, code formatting, and wording that adheres to on-site cultural norms. We find that mentored questions are substantially improved over non-mentored questions, with average scores increasing by 50%. We provide design implications that challenge how socio-technical communities onboard novices across domains.

ACM Classification Keywords

H.5.3. Group and Organization Interfaces: Computer-supported cooperative work

Author Keywords

social Q&A, collaborative editing, e-mentoring, programming

INTRODUCTION

Building and maintaining active online communities is a difficult and well-documented problem across many community types[4, 15, 23]. For prospective community members, barriers such as learning community norms [24], overcoming technical hurdles [31], and resolving conflict [12] can be harmful to participation. In addition, these barriers may significantly affect people in marginalized groups, such as women and people of color, from fully participating in online communities [14]. This is especially pertinent for online programming communities.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyright for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

CHI 2018, April 21–26, 2018, Montreal, QC, Canada

© 2018 ACM. ISBN 978-1-4503-5620-5/18/04...\$15.00

DOI: <https://doi.org/10.1145/3173574.3174182>

Stack Overflow is the largest online programming community [25]. Each month, over 40 million people visit Stack Overflow, a social Q&A site, to learn about, ask, or answer over 14 million programming questions. Despite great popularity, there is evidence that negative behaviors and malfunctioning community mechanics can have long-term effects on site participation. For example, many questions go unanswered [30], and 90% of accepted answers provided by new users are self-answers. For Stack Overflow, "hostile" criticism and conflict [14, 13] is especially problematic for prospective members. As a result, a user may decide not to ask or answer a question for *fear of negative feedback* [14]. These problems can dissuade novices [31] and women [32] from participating in the community. On the other hand, active community members are interested in preserving community norms: not allowing duplicate questions, off-topic or non-closed questions, or poor quality answers. Community members need a mechanism for helping new users ask better questions, while reducing the hostility and negativity of otherwise well-meaning feedback.

In this paper, we applied theory related to learning and communities of practice to a social Q&A site, by using methods related to mutual engagement and formative feedback to improve novices' questions. Building on design claims for increasing engagement in online communities [19], we created *Help Rooms* with collaborative question drafts to enable novices to receive timely and formative feedback from mentors before posting their questions. Our *Help Rooms* work as follows: when a novice is about to post a question, they are asked if they want additional feedback from a mentor. If the novice responds positively, they are redirected to a room with a mentor who can help them edit their question. The mentor offers advice on how to phrase and ask their question so that it can be well received by the Stack Overflow community.

In a one-month online study, we implemented our mechanism for mentored question asking on Stack Overflow, and enabled 271 novices to receive help with their questions. As a result, we found that mentored questions were substantially improved over non-mentored questions. Average scores increased by 50%, resulting in fewer off-topic, deleted, and poor questions. Overall, for mentored questions, there was an increase in the amount of *good* questions asked, and reduction of *bad* questions asked by Stack Overflow standards. Novices surveyed agreed that they feel more comfortable posting on Stack Overflow after their participation (median = 4 on a 5 point Likert

"We don't do that here"

- Question phrasing
- Formatting posts
- Community triage
- Question framing
- Community culture of asking

is this an appropriate place for your Q?

"We Don't Do That Here": How Collaborative Editing with Mentors Improves Engagement in Social Q&A Communities

Denae Ford¹, Kristina Lustig², Jeremy Banks³, Chris Parnin¹

¹North Carolina State University, Raleigh, NC, USA

²Stack Exchange, Inc., New York, NY, USA

{dford3, cjparnin}@ncsu.edu, klustig@stackoverflow.com, _@jeremy.ca

ABSTRACT

Online question-and-answer (Q&A) communities like Stack Overflow have norms that are not obvious to novice users. Novices create and post programming questions without feedback, and the community enforces site norms through public downvoting and commenting. This can leave novices discouraged from further participation. We deployed a month-long, just-in-time mentorship program to Stack Overflow in which we redirected novices in the process of asking a question to an on-site *Help Room*. There, novices received feedback on their question drafts from experienced Stack Overflow mentors. We present examples and discussion of various question improvements including: question context, code formatting, and wording that adheres to on-site cultural norms. We find that mentored questions are substantially improved over non-mentored questions, with average scores increasing by 50%. We provide design implications that challenge how socio-technical communities onboard novices across domains.

ACM Classification Keywords

H.5.3. Group and Organization Interfaces: Computer-supported cooperative work

Author Keywords

social Q&A, collaborative editing, e-mentoring, programming

INTRODUCTION

Building and maintaining active online communities is a difficult and well-documented problem across many community types [4, 15, 23]. For prospective community members, barriers such as learning community norms [24], overcoming technical hurdles [31], and resolving conflict [12] can be harmful to participation. In addition, these barriers may significantly affect people in marginalized groups, such as women and people of color, from fully participating in online communities [14]. This is especially pertinent for online programming communities.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyright for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

CHI 2018, April 21–26, 2018, Montreal, QC, Canada

© 2018 ACM. ISBN 978-1-4503-5620-5/18/04...\$15.00

DOI: <https://doi.org/10.1145/3173574.3174182>

Stack Overflow is the largest online programming community [25]. Each month, over 40 million people visit Stack Overflow, a social Q&A site, to learn about, ask, or answer over 14 million programming questions. Despite great popularity, there is evidence that negative behaviors and malfunctioning community mechanics can have long-term effects on site participation. For example, many questions go unanswered [30], and 90% of accepted answers provided by new users are self-answers. For Stack Overflow, "hostile" criticism and conflict [14, 13] is especially problematic for prospective members. As a result, a user may decide not to ask or answer a question for *fear of negative feedback* [14]. These problems can dissuade novices [31] and women [32] from participating in the community. On the other hand, active community members are interested in preserving community norms: not allowing duplicate questions, off-topic or non-closed questions, or poor quality answers. Community members need a mechanism for helping new users ask better questions, while reducing the hostility and negativity of otherwise well-meaning feedback.

In this paper, we applied theory related to learning and communities of practice to a social Q&A site, by using methods related to mutual engagement and formative feedback to improve novices' questions. Building on design claims for increasing engagement in online communities [19], we created *Help Rooms* with collaborative question drafts to enable novices to receive timely and formative feedback from mentors before posting their questions. Our *Help Rooms* work as follows: when a novice is about to post a question, they are asked if they want additional feedback from a mentor. If the novice responds positively, they are redirected to a room with a mentor who can help them edit their question. The mentor offers advice on how to phrase and ask their question so that it can be well received by the Stack Overflow community.

In a one-month online study, we implemented our mechanism for mentored question asking on Stack Overflow, and enabled 271 novices to receive help with their questions. As a result, we found that mentored questions were substantially improved over non-mentored questions. Average scores increased by 50%, resulting in fewer off-topic, deleted, and poor questions. Overall, for mentored questions, there was an increase in the amount of *good* questions asked, and reduction of *bad* questions asked by Stack Overflow standards. Novices surveyed agreed that they feel more comfortable posting on Stack Overflow after their participation (median = 4 on a 5-point Likert

"We don't do that here"

- Question phrasing
- Formatting posts
- Community triage
- Question framing
- Community culture of asking

clarity, research of problem, context

"We Don't Do That Here": How Collaborative Editing with Mentors Improves Engagement in Social Q&A Communities

Denae Ford¹, Kristina Lustig², Jeremy Banks², Chris Parnin¹

¹North Carolina State University, Raleigh, NC, USA

²Stack Exchange, Inc., New York, NY, USA

{dford3, cjparnin}@ncsu.edu, klustig@stackoverflow.com, _@jeremy.ca

ABSTRACT

Online question-and-answer (Q&A) communities like Stack Overflow have norms that are not obvious to novice users. Novices create and post programming questions without feedback, and the community enforces site norms through public downvoting and commenting. This can leave novices discouraged from further participation. We deployed a month-long, just-in-time mentorship program to Stack Overflow in which we redirected novices in the process of asking a question to an on-site *Help Room*. There, novices received feedback on their question drafts from experienced Stack Overflow mentors. We present examples and discussion of various question improvements including: question context, code formatting, and wording that adheres to on-site cultural norms. We find that mentored questions are substantially improved over non-mentored questions, with average scores increasing by 50%. We provide design implications that challenge how socio-technical communities onboard novices across domains.

ACM Classification Keywords

H.5.3. Group and Organization Interfaces: Computer-supported cooperative work

Author Keywords

social Q&A, collaborative editing, e-mentoring, programming

INTRODUCTION

Building and maintaining active online communities is a difficult and well-documented problem across many community types [4, 15, 23]. For prospective community members, barriers such as learning community norms [24], overcoming technical hurdles [31], and resolving conflict [12] can be harmful to participation. In addition, these barriers may significantly affect people in marginalized groups, such as women and people of color, from fully participating in online communities [14]. This is especially pertinent for online programming communities.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyright for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

CHI 2018, April 21–26, 2018, Montreal, QC, Canada

© 2018 ACM. ISBN 978-1-4503-5630-5/18/04...\$15.00

DOI: <https://doi.org/10.1145/3173574.3174182>

Stack Overflow is the largest online programming community [25]. Each month, over 40 million people visit Stack Overflow, a social Q&A site, to learn about, ask, or answer over 14 million programming questions. Despite great popularity, there is evidence that negative behaviors and malfunctioning community mechanics can have long-term effects on site participation. For example, many questions go unanswered [30], and 90% of accepted answers provided by new users are self-answers. For Stack Overflow, "hostile" criticism and conflict [14, 13] is especially problematic for prospective members. As a result, a user may decide not to ask or answer a question for *fear of negative feedback* [14]. These problems can dissuade novices [31] and women [32] from participating in the community. On the other hand, active community members are interested in preserving community norms: not allowing duplicate questions, off-topic or non-closed questions, or poor quality answers. Community members need a mechanism for helping new users ask better questions, while reducing the hostility and negativity of otherwise well-meaning feedback.

In this paper, we applied theory related to learning and communities of practice to a social Q&A site, by using methods related to mutual engagement and formative feedback to improve novices' questions. Building on design claims for increasing engagement in online communities [19], we created *Help Rooms* with collaborative question drafts to enable novices to receive timely and formative feedback from mentors before posting their questions. Our *Help Rooms* work as follows: when a novice is about to post a question, they are asked if they want additional feedback from a mentor. If the novice responds positively, they are redirected to a room with a mentor who can help them edit their question. The mentor offers advice on how to phrase and ask their question so that it can be well received by the Stack Overflow community.

In a one-month online study, we implemented our mechanism for mentored question asking on Stack Overflow, and enabled 271 novices to receive help with their questions. As a result, we found that mentored questions were substantially improved over non-mentored questions. Average scores increased by 50%, resulting in fewer off-topic, deleted, and poor questions. Overall, for mentored questions, there was an increase in the amount of *good* questions asked, and reduction of *bad* questions asked by Stack Overflow standards. Novices surveyed agreed that they feel more comfortable posting on Stack Overflow after their participation (median = 4 on a 5-point Likert

"We don't do that here"

- Question phrasing
- Formatting posts
- Community triage
- Question framing
- Community culture of asking

You also might want to edit out the “Thank you!” at the end. I know it seems polite, but people object to it on Stack Overflow.

"We Don't Do That Here": How Collaborative Editing with Mentors Improves Engagement in Social Q&A Communities

Denae Ford¹, Kristina Lustig², Jeremy Banks², Chris Parnin¹

¹North Carolina State University, Raleigh, NC, USA

²Stack Exchange, Inc., New York, NY, USA

{dford3, cjparnin}@ncsu.edu, klustig@stackoverflow.com, _@jeremy.ca

ABSTRACT

Online question-and-answer (Q&A) communities like Stack Overflow have norms that are not obvious to novice users. Novices create and post programming questions without feedback, and the community enforces site norms through public downvoting and commenting. This can leave novices discouraged from further participation. We deployed a month-long, just-in-time mentorship program to Stack Overflow in which we redirected novices in the process of asking a question to an on-site *Help Room*. There, novices received feedback on their question drafts from experienced Stack Overflow mentors. We present examples and discussion of various question improvements including: question context, code formatting, and wording that adheres to on-site cultural norms. We find that mentored questions are substantially improved over non-mentored questions, with average scores increasing by 50%. We provide design implications that challenge how socio-technical communities onboard novices across domains.

ACM Classification Keywords

H.5.3. Group and Organization Interfaces: Computer-supported cooperative work

Author Keywords

social Q&A, collaborative editing, e-mentoring, programming

INTRODUCTION

Building and maintaining active online communities is a difficult and well-documented problem across many community types[4, 15, 23]. For prospective community members, barriers such as learning community norms [24], overcoming technical hurdles [31], and resolving conflict [12] can be harmful to participation. In addition, these barriers may significantly affect people in marginalized groups, such as women and people of color, from fully participating in online communities [14]. This is especially pertinent for online programming communities.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyright for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

CHI 2018, April 21–26, 2018, Montreal, QC, Canada

© 2018 ACM. ISBN 978-1-4503-5620-5/18/04...\$15.00

DOI: <https://doi.org/10.1145/3173574.3174182>

Stack Overflow is the largest online programming community [25]. Each month, over 40 million people visit Stack Overflow, a social Q&A site, to learn about, ask, or answer over 14 million programming questions. Despite great popularity, there is evidence that negative behaviors and malfunctioning community mechanics can have long-term effects on site participation. For example, many questions go unanswered [30], and 90% of accepted answers provided by new users are self-answers. For Stack Overflow, “hostile” criticism and conflict [14, 13] is especially problematic for prospective members. As a result, a user may decide not to ask or answer a question for *fear of negative feedback* [14]. These problems can dissuade novices [31] and women [32] from participating in the community. On the other hand, active community members are interested in preserving community norms: not allowing duplicate questions, off-topic or non-closed questions, or poor quality answers. Community members need a mechanism for helping new users ask better questions, while reducing the hostility and negativity of otherwise well-meaning feedback.

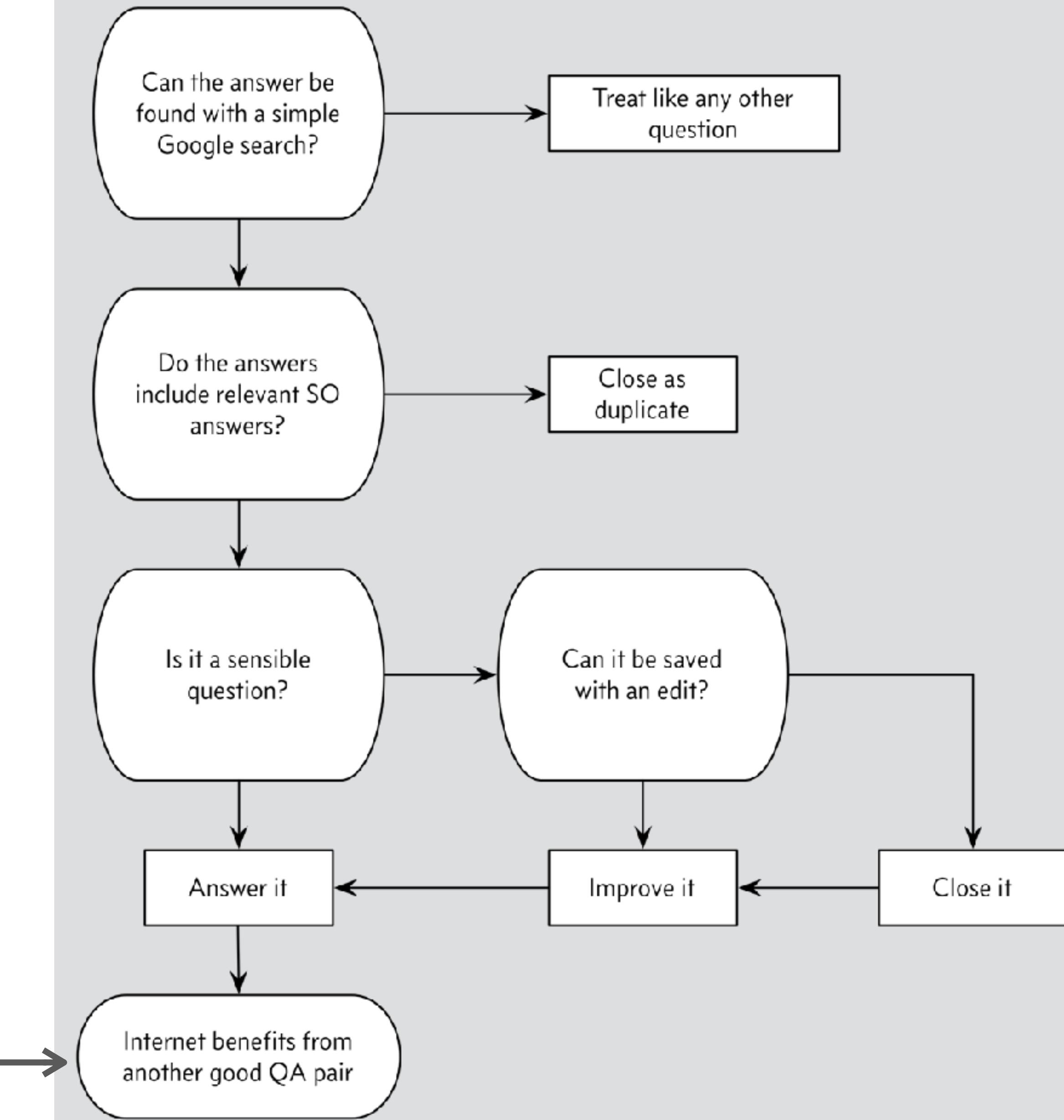
In this paper, we applied theory related to learning and communities of practice to a social Q&A site, by using methods related to mutual engagement and formative feedback to improve novices’ questions. Building on design claims for increasing engagement in online communities [19], we created *Help Rooms* with collaborative question drafts to enable novices to receive timely and formative feedback from mentors before posting their questions. Our *Help Rooms* work as follows: when a novice is about to post a question, they are asked if they want additional feedback from a mentor. If the novice responds positively, they are redirected to a room with a mentor who can help them edit their question. The mentor offers advice on how to phrase and ask their question so that it can be well received by the Stack Overflow community.

In a one-month online study, we implemented our mechanism for mentored question asking on Stack Overflow, and enabled 271 novices to receive help with their questions. As a result, we found that mentored questions were substantially improved over non-mentored questions. Average scores increased by 50%, resulting in fewer off-topic, deleted, and poor questions. Overall, for mentored questions, there was an increase in the amount of *good* questions asked, and reduction of *bad* questions asked by Stack Overflow standards. Novices surveyed agreed that they feel more comfortable posting on Stack Overflow after their participation (median = 4 on a 5 point Likert

How should you respond to RTFM questions?



goal = good QA pair



10 simple rules for getting help from online scientific communities

1. Do not be afraid to ask a question
2. State the question clearly
3. New to a mailing list? Learn the established customs before posting
4. Do not ask what has already been answered
5. Always use a good title
6. Do your homework before posting
7. Proofread your post and write in correct English
8. Be courteous to other forum members
9. Remember that the archive of your discussion can be useful to other people
10. Give back to the community



The screenshot shows the Discourse website homepage with a large banner reading "Civilized discussion for your community". Below the banner is a "Try it FREE" button. The main content area displays a "Your Community" feed with various posts. A modal window titled "Your Community" is overlaid on the feed, showing a list of posts with user profiles and interaction counts.

Your Community

Sign Up Log In Search

Post Title	User Profile	Praise	Replies	Views	Date
Net Neutrality Day of Action	Profile 1, Profile 2, Profile 3, Profile 4	4	878	Jul '17	
Sharing our experience changing the Discourse UI	Profile 1, Profile 2, Profile 3, Profile 4, Profile 5	6	2.4k	Jul '17	
A measure of success	Profile 1	0	763		
Discourse app: just GREAT	Profile 1	0	830		
The community platform of champions	Profile 1	0	890		
Loving Discourse so far	Profile 1	0	807		
Discourse for discussions: superiority to e.g. GitHub	Profile 1	0	1.2k		
Discourse running great on forum with a few hundred users	Profile 1	0	823		
Minimally interesting to suggest today - [REDACTED]	Profile 1, Profile 2, Profile 3, Profile 4	0	1.7k		

Your Community Log In Search

Post Title	User Profile	Praise	Replies	Views	Date
Discourse is Best Forum Solution in the 2017 CMS Critic Awards	Profile 1	7		Nov '17	
Converting, combining, migrating - Yes!	Profile 1	4		Nov '17	
Successful Migration	Profile 1	0		Oct '17	
Thanks to discourse	Profile 1	6		Oct '17	
Post your Discourse T-shirt pics!	Profile 1	45		Oct '17	
What an amazon platform...	Profile 1	0			

<https://www.discourse.org/>

RStudio Community

https://community.rstudio.com

Sign Up Log In

Latest Categories Top

Topic	Category	Users	Replies	Views	Activity
<input checked="" type="checkbox"/> Welcome to the RStudio Community! Welcome to community.rstudio.com — we're glad to have you! This welcome page will give you some advice on how to get the most out of the site if you're getting or giving help. We want this to be a friendly, inclusive com... read more	meta		0	2.8k	Jul 22
<input type="checkbox"/> Integrating shiny apps shiny	shiny		1	34	16m
<input type="checkbox"/> Create pdf from Rnw: missing just one font character from eastern European language	R Markdown		4	39	30m
<input type="checkbox"/> UNC pathway when downloading packages ide ide-issue	RStudio IDE		0	6	39m
<input type="checkbox"/> Are packrat bundles really portable across different platforms?	R Admins		12	592	1h
<input type="checkbox"/> /usr/bin/env no such file or directory terminal rstudioserver	R Admins		1	10	1h
<input type="checkbox"/> Deploying APP on shinyapps.io which keeps asking for ggplot2 ggplot2 shinyappsi0 package-installation	shiny		13	92	1h
<input type="checkbox"/> when I installed R Studio, the packages were visible, but not seeing packages not seeing anymore, how to get back packages in rstudio? ide ide-issue	RStudio IDE		0	5	2h
<input type="checkbox"/> How to read Netcdf files	General		8	58	4h

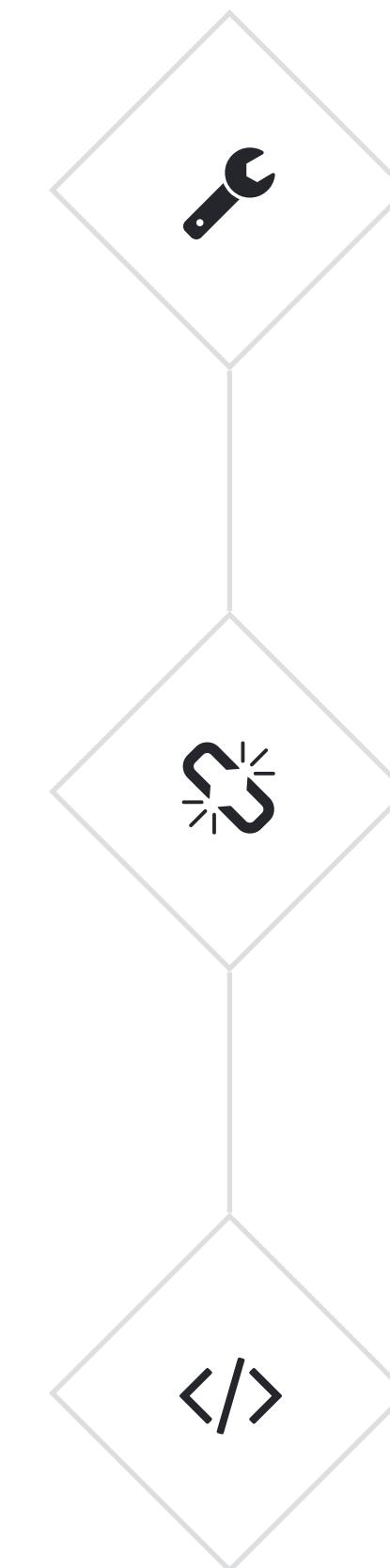
The art of the question

The most useless problem statement that one can face is “it doesn’t work”, yet we seem to get it far too often.

- Thiago Maciera



the anatomy of an issue



PROBLEM DESCRIPTION

EXPECTED BEHAVIOUR

MINIMAL REPRODUCIBLE EXAMPLE

the anatomy of an issue

"Creating an issue template for your repository" - GitHub Help <<https://help.github.com/articles/creating-an-issue-template-for-your-repository/>>

Error Information: Packages missing

Description of issue - [REDACTED]

Steps taken so far - i tried and tried..

System Information:

- RStudio Edition: (Desktop or Server)
- RStudio Version:
- OS Version:
- R Version:

Also:

- RStudio diagnostics report:
- Your sessionInfo():
- RStudio crash report:
- RStudio application log files:

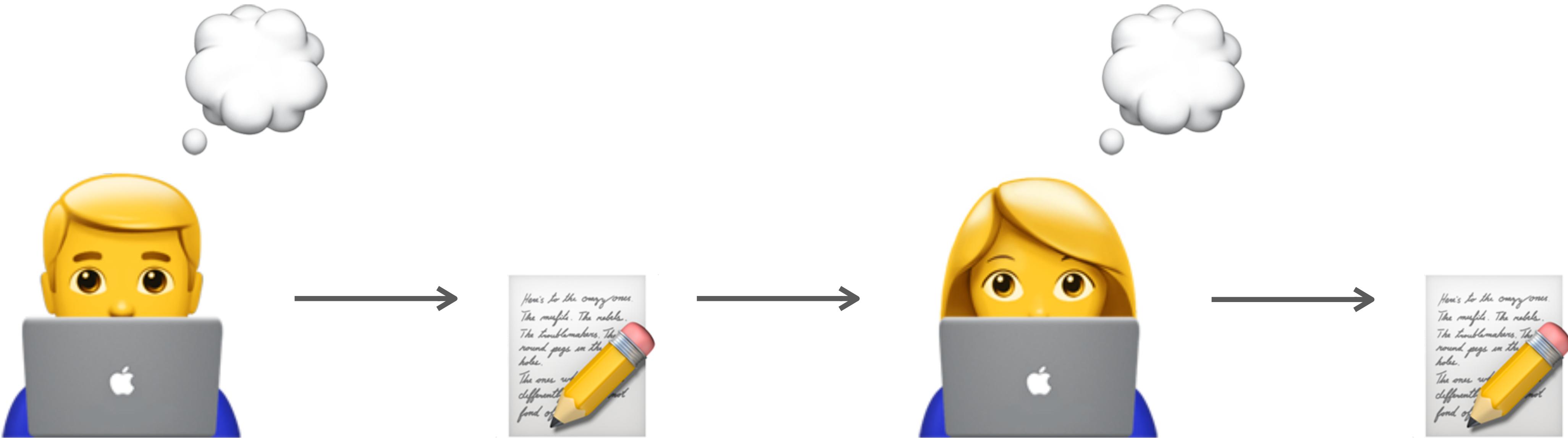
From [Troubleshooting Guide: Using RStudio](#)

They're not *wrong*

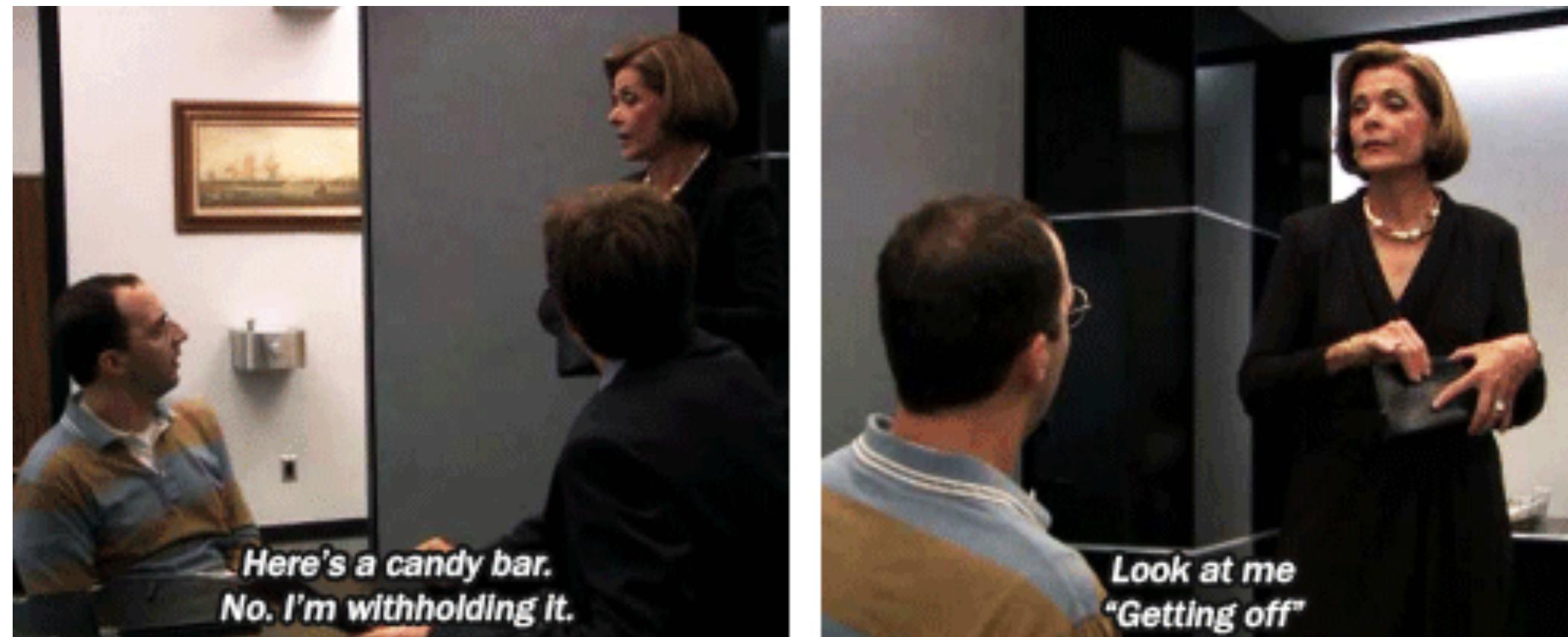
“It is impossible to speak in such a way that you cannot be misunderstood.”

– Karl Popper

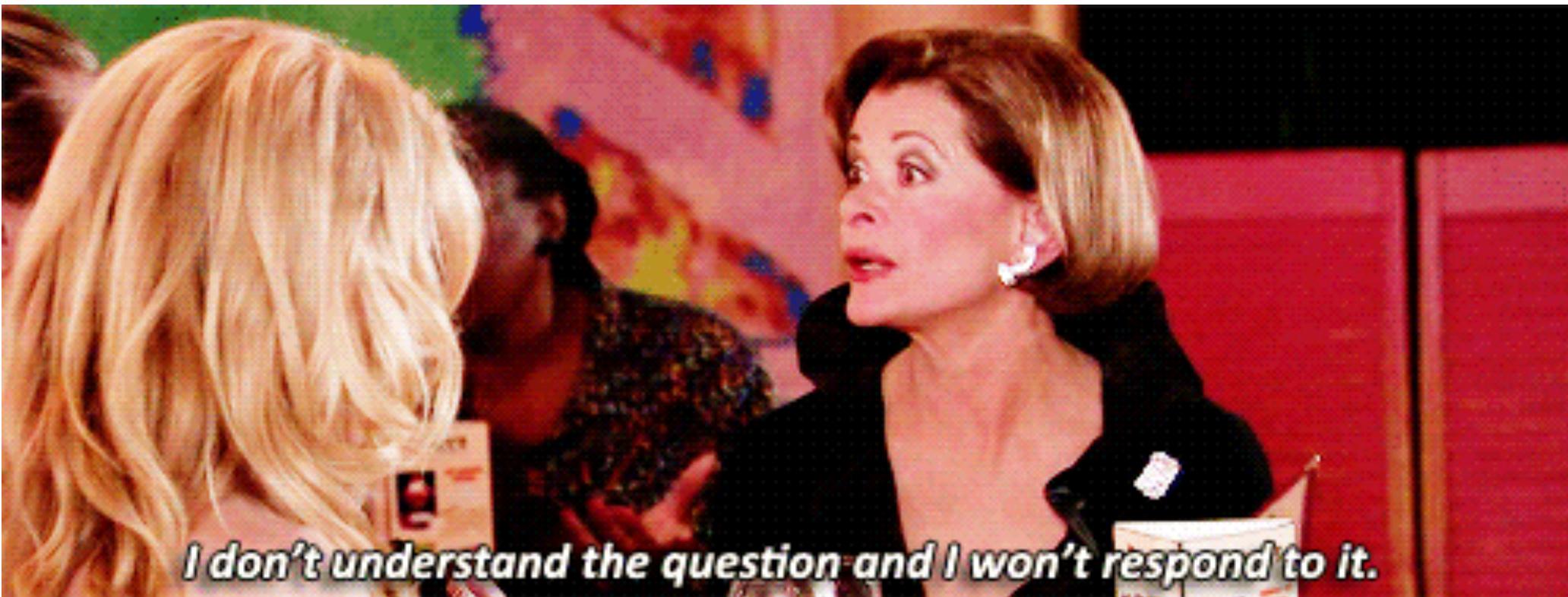
Writing prose about code...



Writing prose about code...

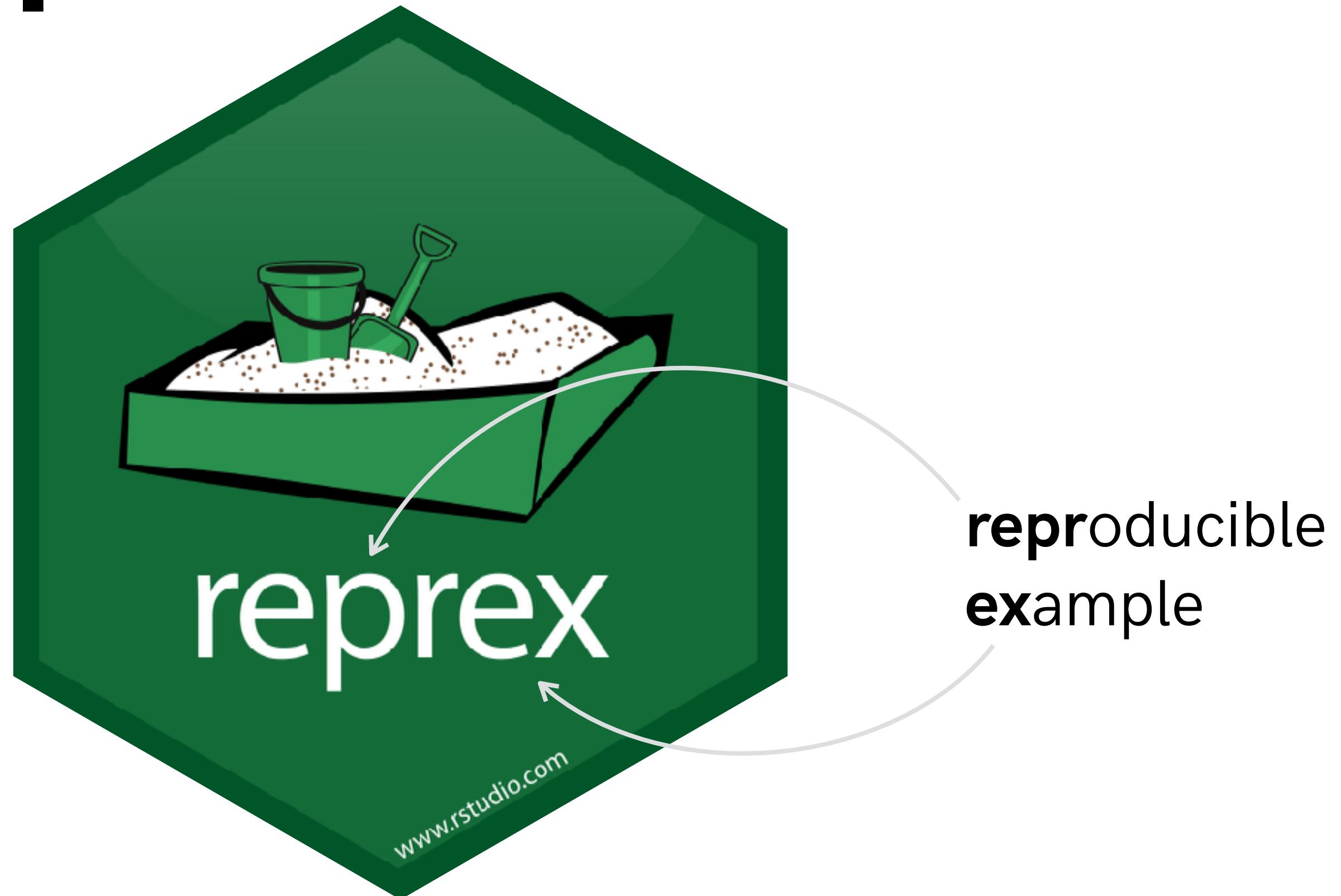


Writing prose about code...



I don't understand the question and I won't respond to it.

the magic of reprex



reprex raison d'être



Keys to reprex-cellence

- ✓ Code that **actually runs**
- ✓ Code that **doesn't have to be run**
- ✓ Code that **can be easily run**

The screenshot shows an RStudio interface with the following components:

- Left Panel (Code Editor):** An R script titled "2017-01-03-reprex-magic.Rmd" containing the following code:

```
1 library(visdat)
2 
3 vis_miss(airquality)
4 
5 library(ggplot2)
6 
7 ggplot(airquality,
8       aes(x = Ozone,
9            y = Solar.R)) +
10   geom_point()
11 
12 library(naniar)
13 
14 ggplot(airquality,
15       aes(x = Ozone,
16            y = Solar.R)) +
17   geom_missing_point()
```
- Top Bar:** Shows tabs for "2017-01-03-reprex-magic.Rmd", "Untitled1*", and "Untitled2*". It also includes "Run", "Source", and "Console" buttons.
- Console:** Displays the output of running the script:

```
> reprex::reprex()
Rendered reprex ready on the clipboard.
> reprex::reprex()
Rendered reprex ready on the clipboard.

Restarting R session...

> reprex::reprex()
Rendered reprex ready on the clipboard.
>
```
- Bottom Panel (Viewer):** Shows the rendered output of the code:

```
vis_miss(airquality)
#> Error in eval(expr, envir, enclos): could not find function "vis_miss"

ggplot(airquality,
       aes(x = Ozone,
            y = Solar.R)) +
  geom_point()
#> Error in eval(expr, envir, enclos): could not find function "ggplot"

ggplot(airquality,
       aes(x = Ozone,
            y = Solar.R)) +
  geom_missing_point()
#> Error in eval(expr, envir, enclos): could not find function "ggplot"
```

Source: Nick Tierney. "Magic reprex." 2017-01-11 <<http://www.njtierney.com/post/2017/01/11/magic-reprex/>>

The screenshot shows the RStudio interface. On the left, the 'Script' tab of the 'Untitled1' file is open, displaying the following R code:

```
1 library(ggplot2)
2
3 df <- data.frame(x = 1)
4
5 ggplot(df, aes(x, x)) + geom_point() +
6   ggtitle("gjpjQ") +
7   theme(plot.title = element_text(size = 10))
8
9 ggplot(df, aes(x, x)) + geom_point() +
10  ggtitle("gjpjQ") +
11  theme(plot.title = element_text(size = 100))
```

The 'Console' tab is selected in the top right, showing the command prompt: > |

The bottom navigation bar shows tabs for Environment, History, and Connections.

<https://maraaverick.rbind.io/2018/06/reprexcellence/>

The reprex request trifecta



WHAT I'M ASKING YOU TO DO

Make a reproducible example

WHY I'M ASKING YOU TO DO IT

Help me help you — I need your data to do so

HOW YOU CAN DO THE THING

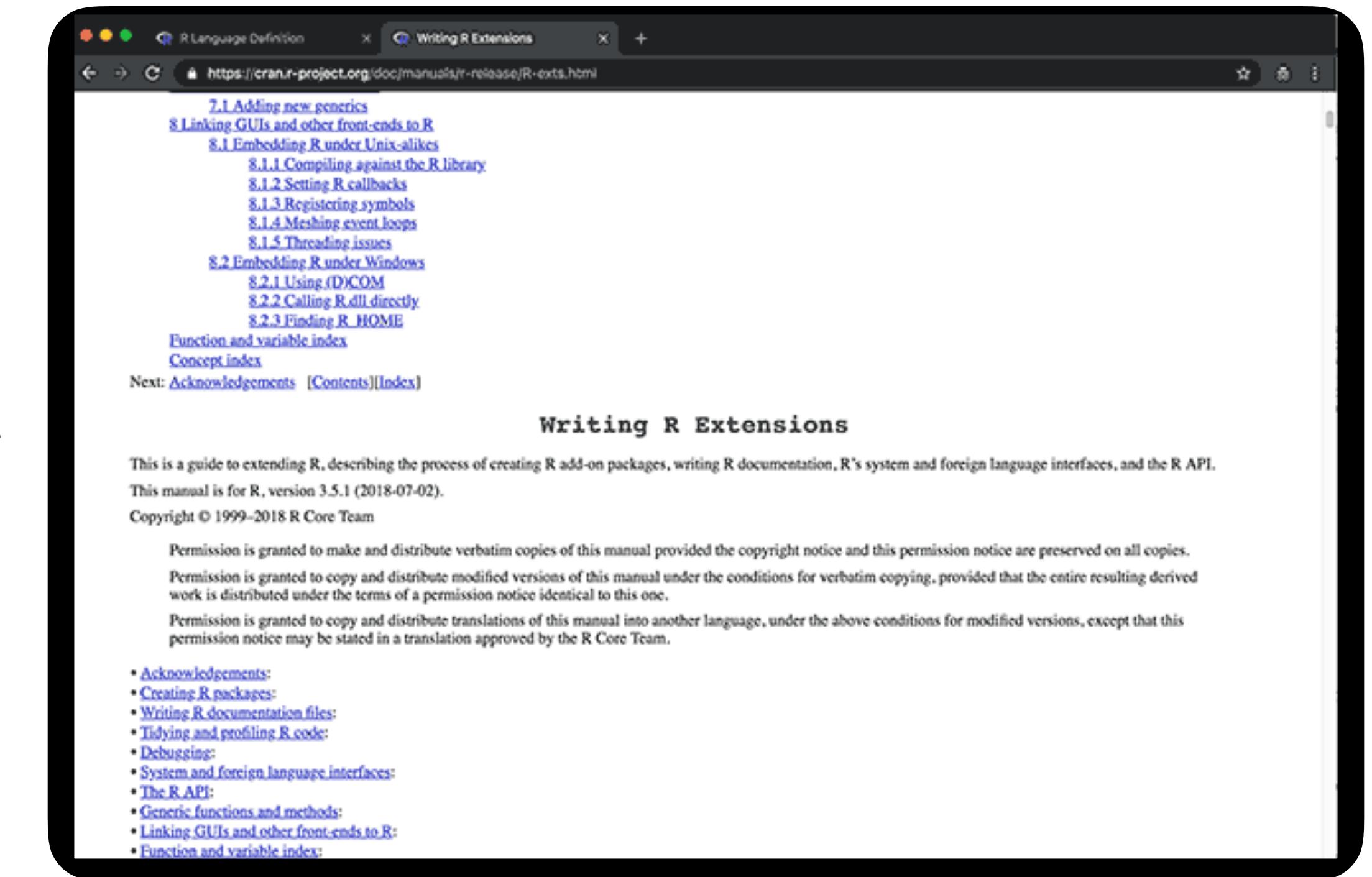
Resources, videos, we've got it all...

RTFM



RTFM

TFM



Guide them through the docs



Tidyr::separate() at second/last occurrence of character

tidyverse

tidy

regex



mara Sustainer

May 20

As you guessed, a regular expression is your best bet here. From the [tidyr separate\(\) function reference](#)

Given either regular expression or a vector of character positions, `separate()` turns a single character column into multiple columns.

[RegExr.com](#) has been my go-to for regular expression testing for a while, but Garrick Aden-Buie recently made an RStudio add-in inspired by the very same, [RegExplain](#), which is a great option for keeping your workflow all in one place:

<https://www.garrickadenbuie.com/project/regexplain/>

Here's how I would do it (which is hacky, but it works); you want to split at the underscore that precedes the day, which is numerical. So, I would use a stringr [look around](#) to say, in effect, "*split at the underscore that is followed by a digit*."

```
suppressPackageStartupMessages(library(tidyverse))
example_data <- data.frame(subject_ID = 1:5,
                           daily_measure_1 = 6:10,
                           daily_measure_2 = 11:15,
                           daily_measure_3 = 16:20,
                           daily_measure_4 = 21:25,
                           daily_measure_5 = 26:30)

example_data %>%
  gather(key = "full_name", value = "value", starts_with("daily_")) %>%
  separate(full_name, c("variable", "day"), sep = "_(?:[0-9])")
```

subject_ID	variable	day	value	
#> 1	1	daily_measure	1	6
#> 2	2	daily_measure	1	7
#> 3	3	daily_measure	1	8
#> 4	4	daily_measure	1	9
#> 5	5	daily_measure	1	10



Tidyr::separate() at second/last occurrence of character



As you guessed, a regular expression is your best bet here. From the `tidyr separate()` function reference 8

Given either regular expression or a vector of character positions, `separate()` turns a single character column into multiple columns.

workflow all in one place:

<https://www.garrickadenbuie.com/project/regexplain/> 30

Here's how I would do it (which is hacky, but it works); you want to split at the underscore that precedes the day, which is numerical. So, I would use a stringr `look around` 18 to say, in effect, "*split at the underscore that is followed by a digit.*"

```
suppressPackageStartupMessages(library(tidyverse))
example_data <- data.frame(subject_ID = 1:5,
                           daily_measure_1 = 6:10,
                           daily_measure_2 = 11:15,
                           daily_measure_3 = 16:20,
                           daily_measure_4 = 21:25,
                           daily_measure_5 = 26:30)
example_data %>%
  gather(key = "full_name", value = "value", starts_with("daily_")) %>%
  separate(full_name, c("variable", "day"), sep = "_(?=[[:digit:]])")
#>   subject_ID   variable day value
#> 1           1 daily_measure  1    6
#> 2           2 daily_measure  1    7
#> 3           3 daily_measure  1    8
#> 4           4 daily_measure  1    9
#> 5           5 daily_measure  1   10
```



Tidyr::separate() at second/last occurrence of character

tidyverse

tidy

regex



mara Sustainer

May 20

As you guessed, a regular expression is your best bet here. From the [tidyr separate\(\) function reference](#)

[RegExr.com](#) has been my go-to for regular expression testing for a while, but Garrick Aden-Buie recently made an RStudio add-in inspired by the very same, [RegExplain](#), which is a great option for keeping your workflow all in one place:

<https://www.garrickadenbuie.com/project/regexplain/>

Here's how I would do it (which is hacky, but it works); you want to split at the underscore that precedes the day, which is numerical. So, I would use a stringr [look around](#) to say, in effect, "*split at the underscore that is followed by a digit.*"

```
daily_measure_1 = 10:20,
daily_measure_4 = 21:25,
daily_measure_5 = 26:30)

example_data %>%
  gather(key = "full_name", value = "value", starts_with("daily_")) %>%
  separate(full_name, c("variable", "day"), sep = "_(?:[0-9])")
```

subject_ID	variable	day	value	
#> 1	1	daily_measure	1	6
#> 2	2	daily_measure	1	7
#> 3	3	daily_measure	1	8
#> 4	4	daily_measure	1	9
#> 5	5	daily_measure	1	10

Contributing to FOSS

WHAT HOLDS PEOPLE BACK?



Contributing to FOSS

WHAT HOLDS PEOPLE BACK?

- “*I can't write code.*”



Contributing to FOSS

WHAT HOLDS PEOPLE BACK?

- “*I can't write code.*”
- “*I'm not really good at this.*”



Contributing to FOSS

WHAT HOLDS PEOPLE BACK?

- “*I can't write code.*”
- “*I'm not really good at this.*”
- “*I'd just be a burden.*”



Contributing to FOSS

WHAT HOLDS PEOPLE BACK?

- “*I can't write code.*”
- “*I'm not really good at this.*”
- “*I'd just be a burden.*”
- “*They already have enough people smarter than me.*”



The newcomer's paradox...

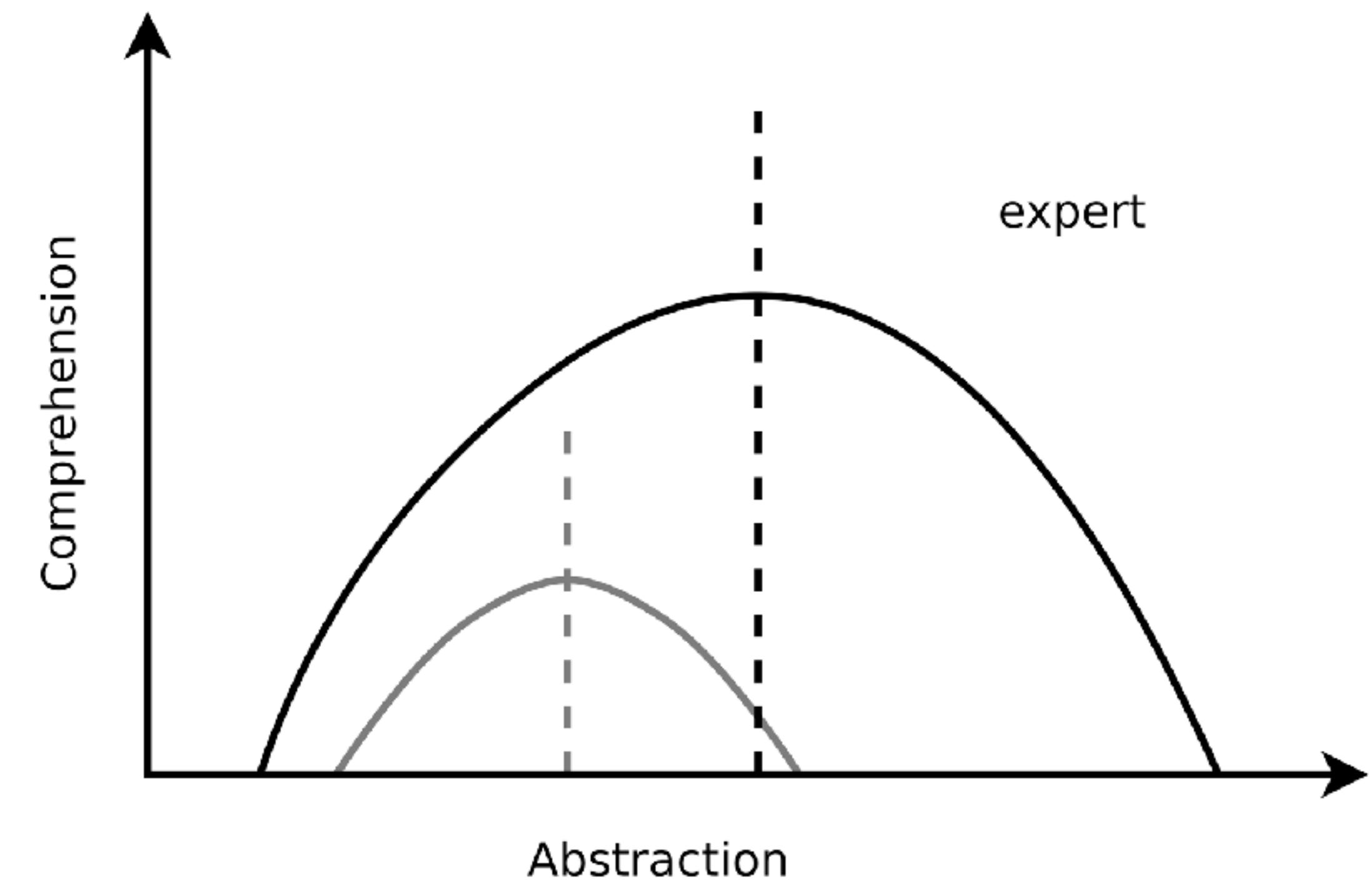
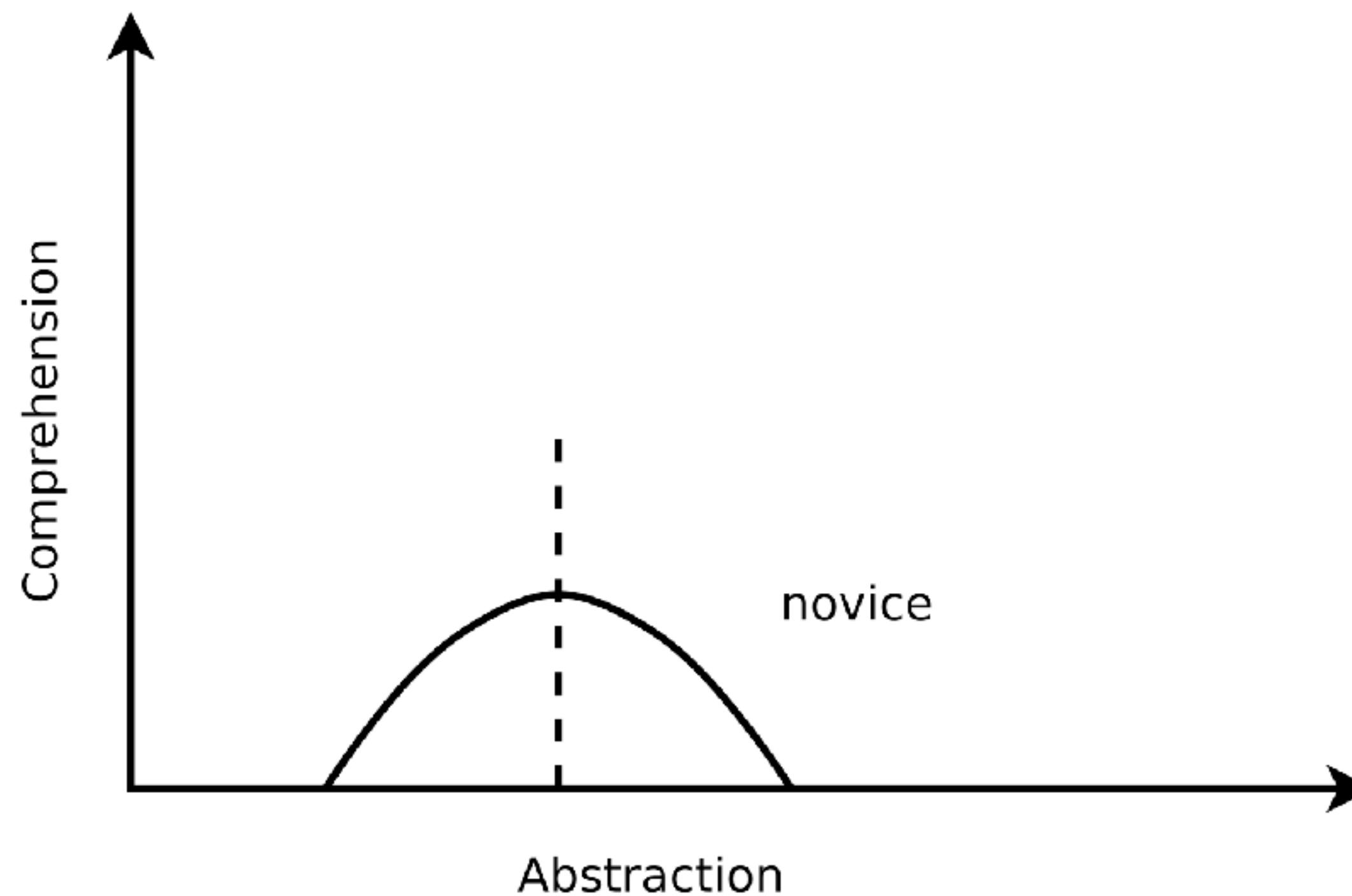


When you ask for help, some friendly soul will no doubt tell you that "it's easy, just do foo, bar and baz." Except for you, it is not easy, there may be no documentation for foo, bar is not doing what it is supposed to be doing and what is this baz thing anyway with its eight disambiguation entries on Wikipedia?

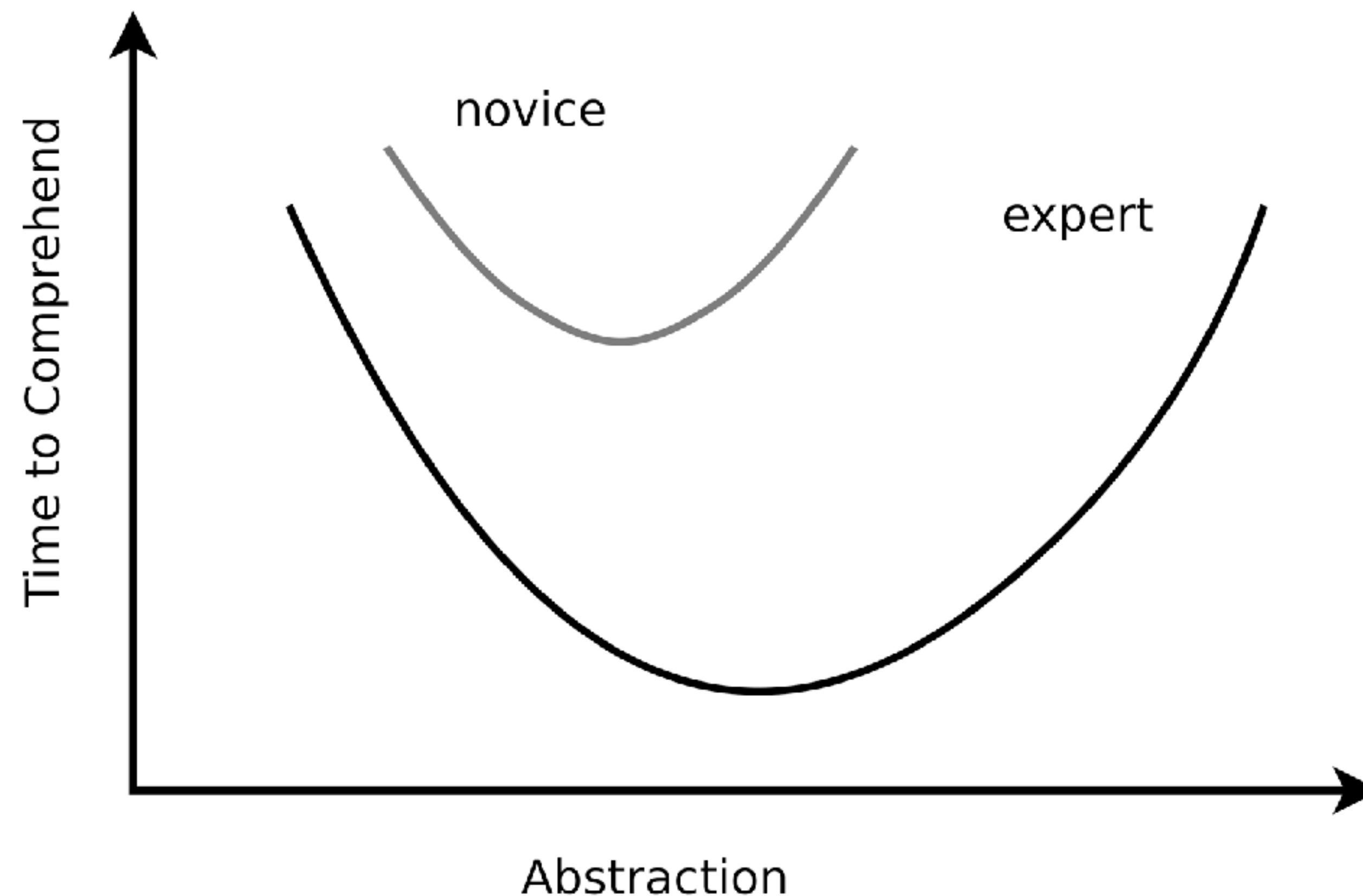
— Leslie Hawthorne

"You'll Eventually Know Everything They've Forgotten." In Open Advice: FOSS: *What We Wish We Had Known When We Started*, edited by Lydia Pintscher, 29–32.

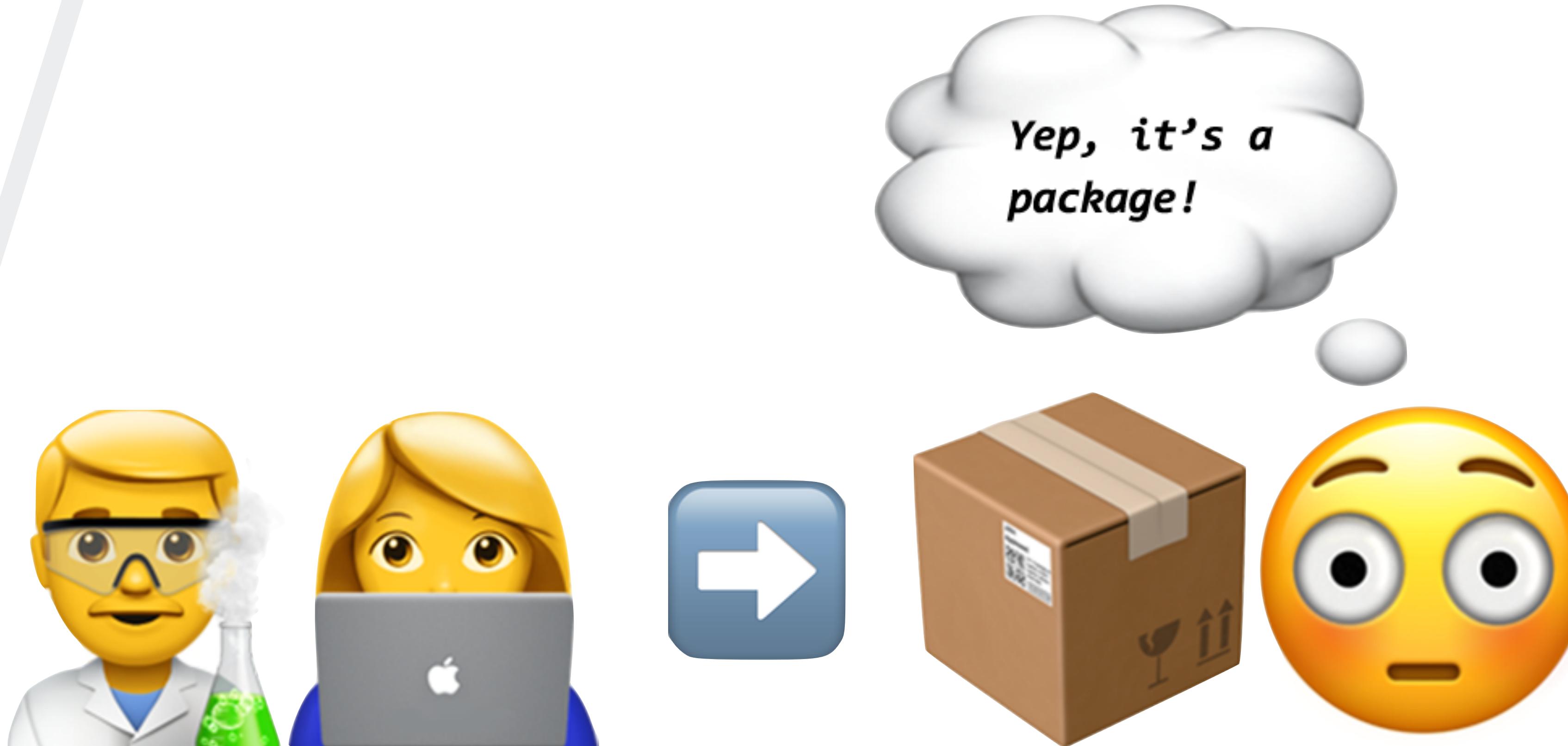
The comprehension comfort curve



Abstraction & comprehension



require(n00bs)



require(n00bs)

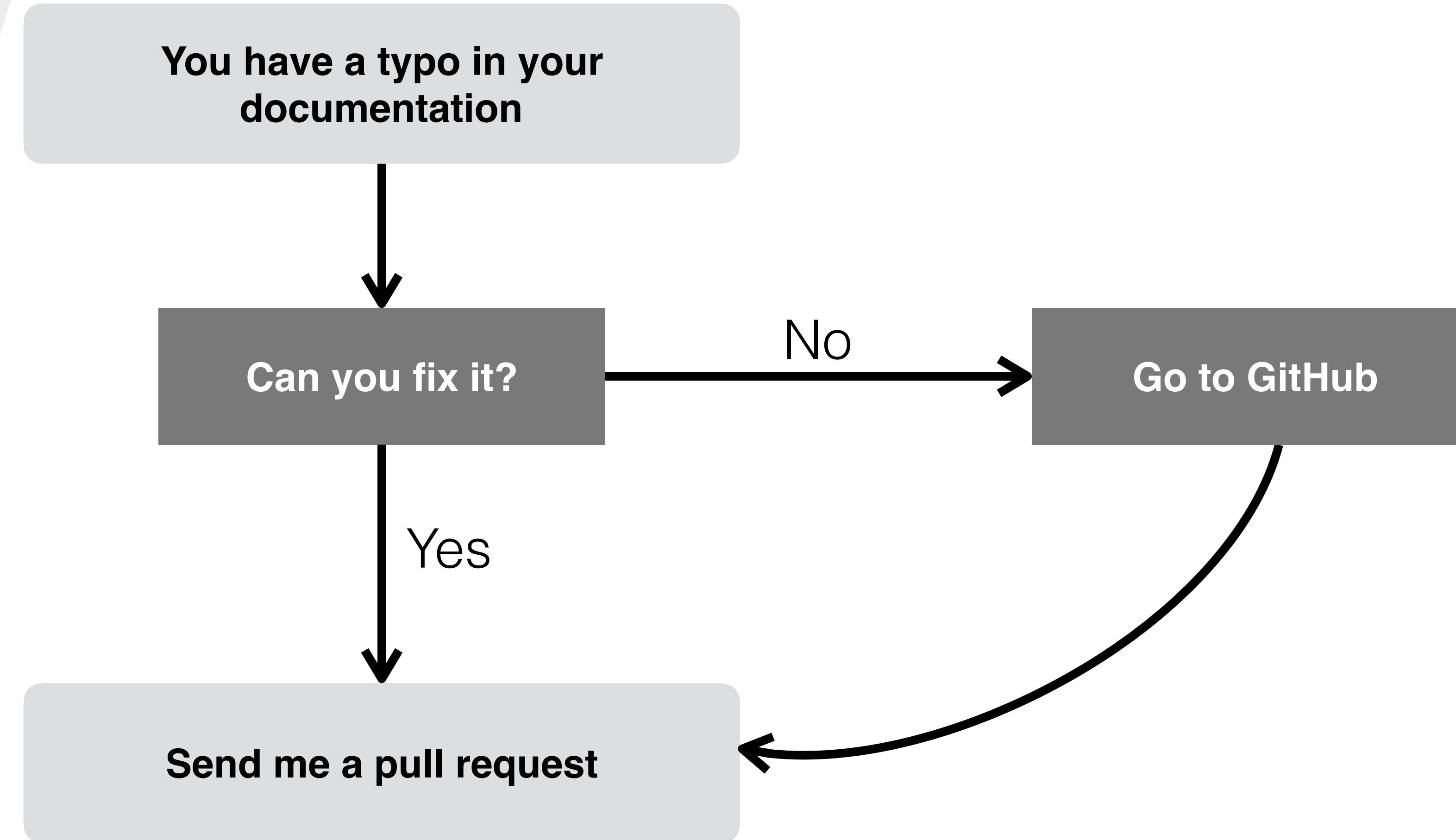
When assigning reviewers to a submission, we aim to pair experienced reviewers with new ones, or reviewers with expertise on a package's programming methods with those experienced in its field of application.



How rOpenSci uses Code Review to Promote Reproducible Science

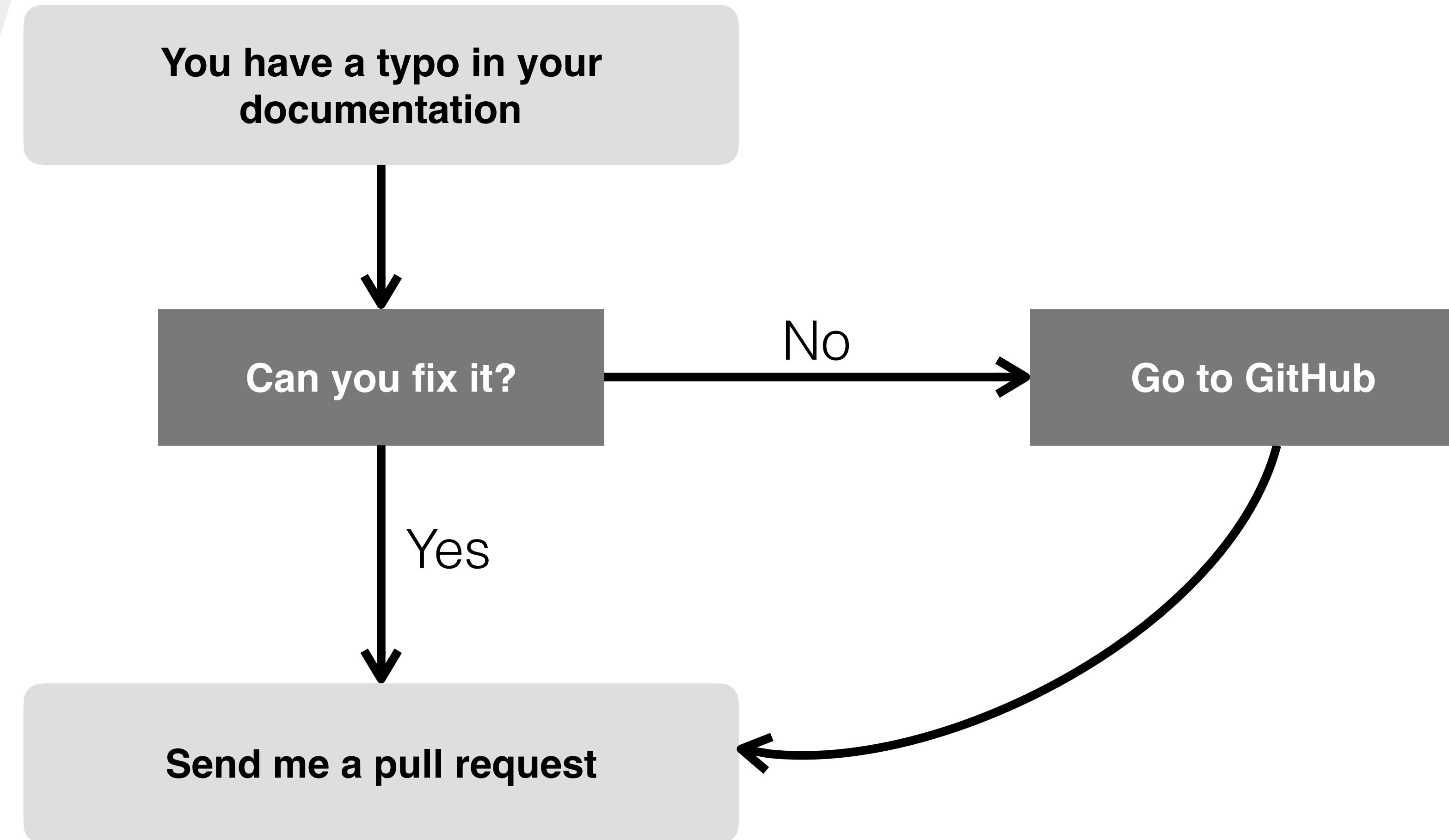
ropensci.org highlighted with Highly

tpyos



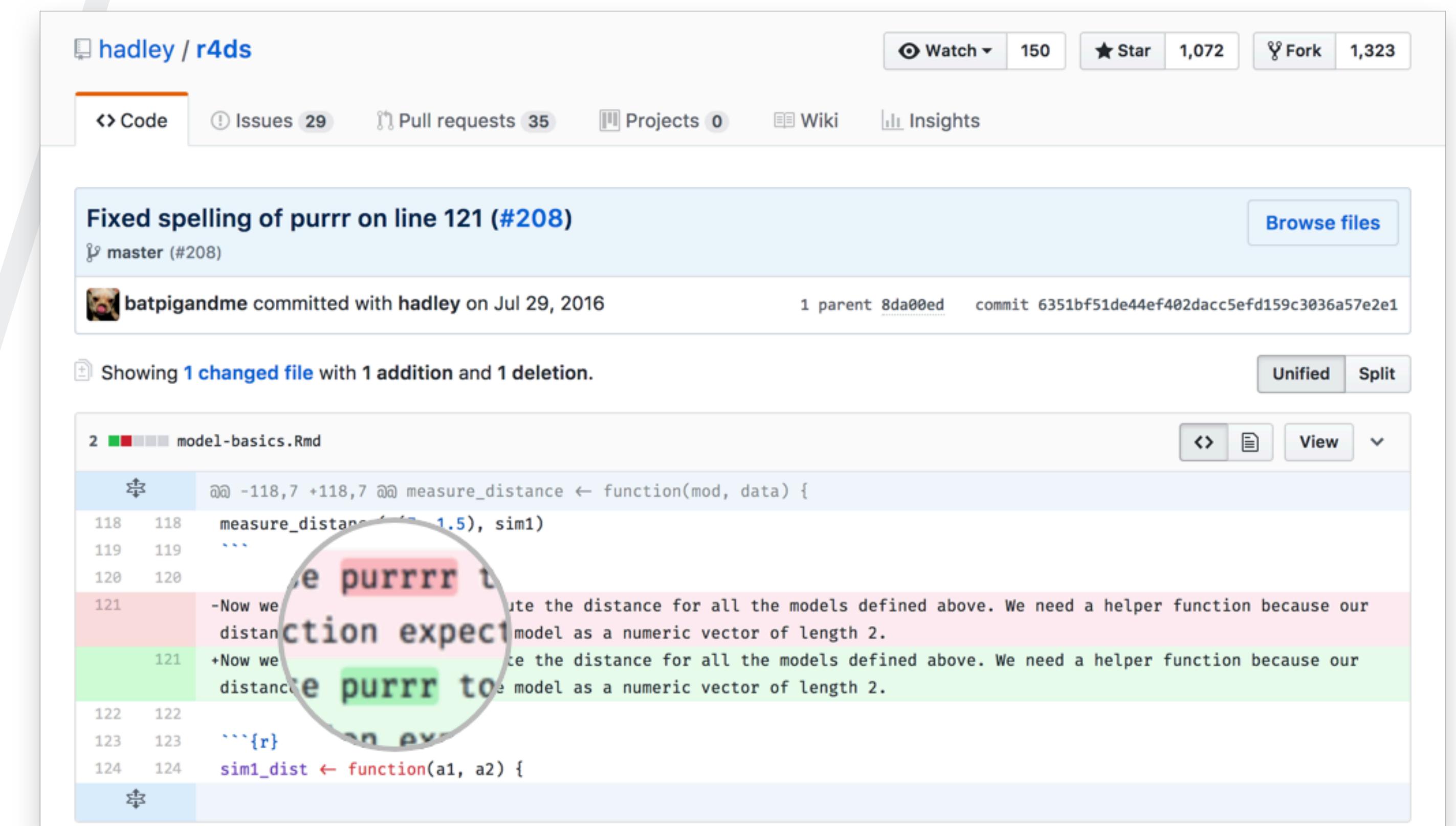
Adapted from: You Do Not Need to Tell Me I Have A Typo in My Documentation by Yihui Xie

typos

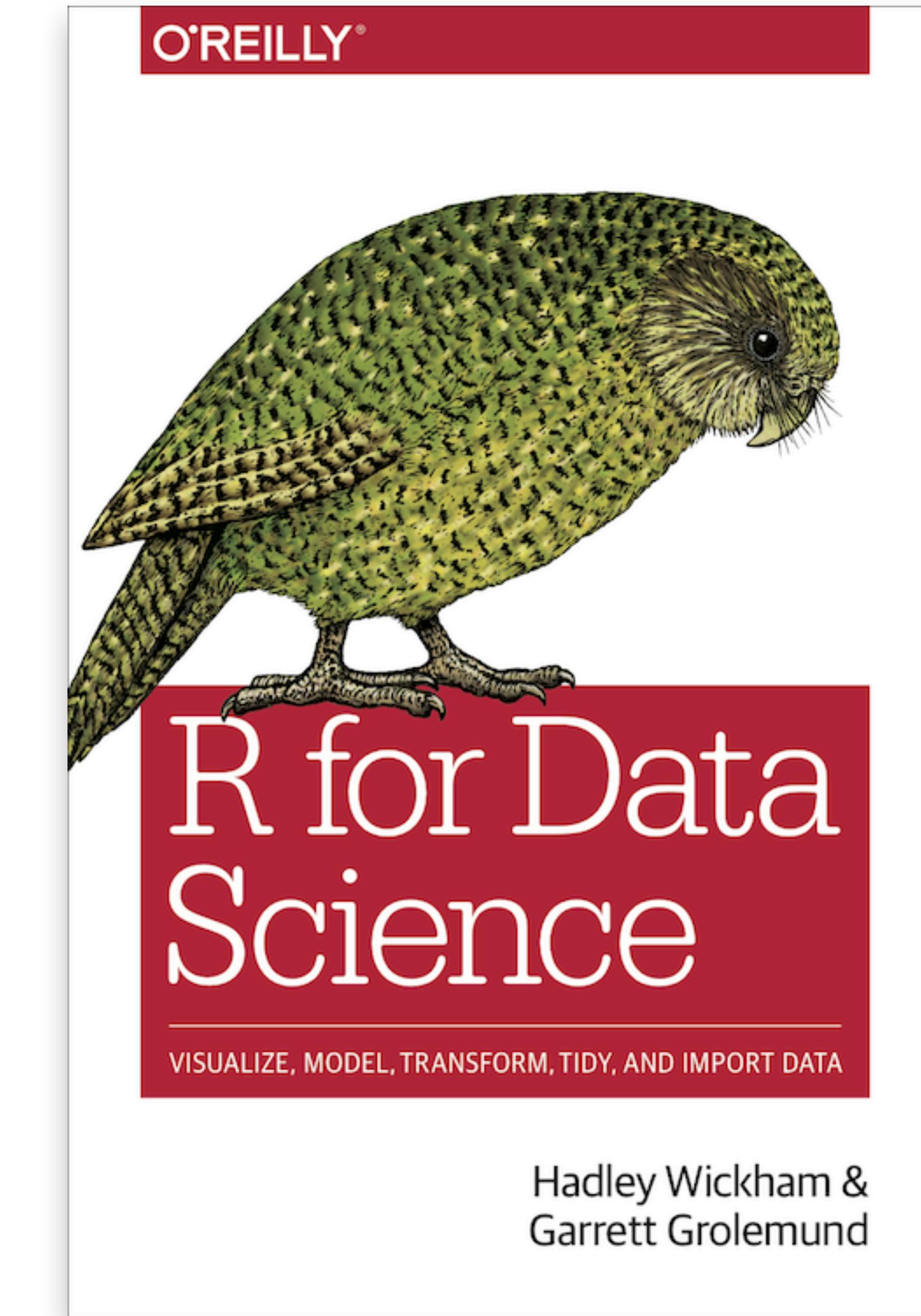


Adapted from: You Do Not Need to Tell Me I Have A Typo in My Documentation by Yihui Xie

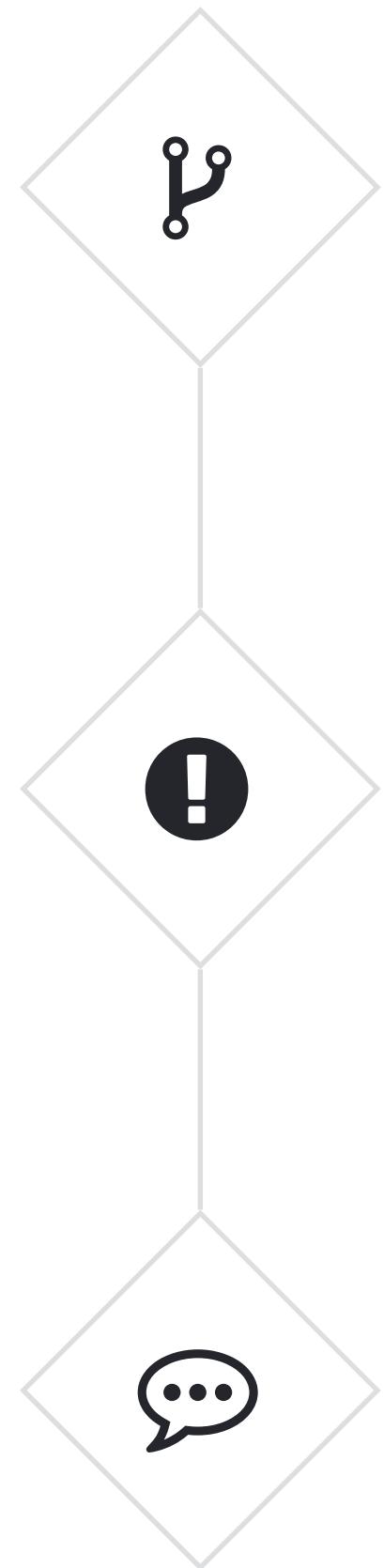
My first “contribution”



My first “contribution”



Ways to contribute



PULL REQUESTS

Contributing code/making fixes.

ISSUES

Identifying a problem, trying your best to isolate its source.

COMMENTS

Help maintainers answer questions, triage issues. Help newcomers learn how to ask better questions (e.g. the art of the reprex).

need help getting started?

Save the date: tidyverse developer day



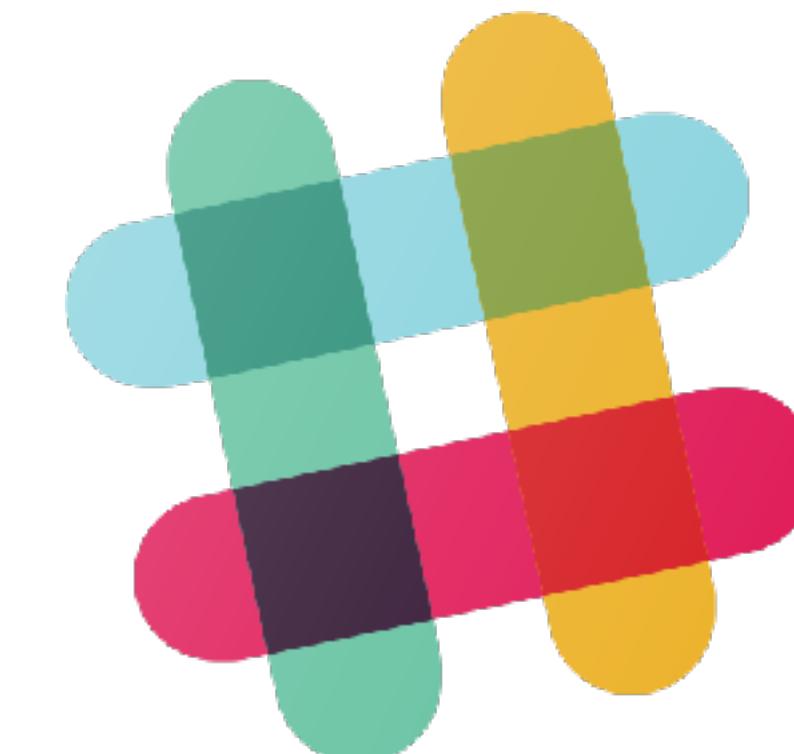
Photo by [rawpixel](#)

Save the date 📅

On the Saturday following [rstudio::conf](#), we'll be holding our first ever tidyverse developer day, and you're invited! So, if you're interested, plan on being in Austin on the 19th of January. The venue is not settled yet, but the intent is to make it convenient for people who've chosen lodging based on the [rstudio::conf](#) location.

Squad goals 🏊

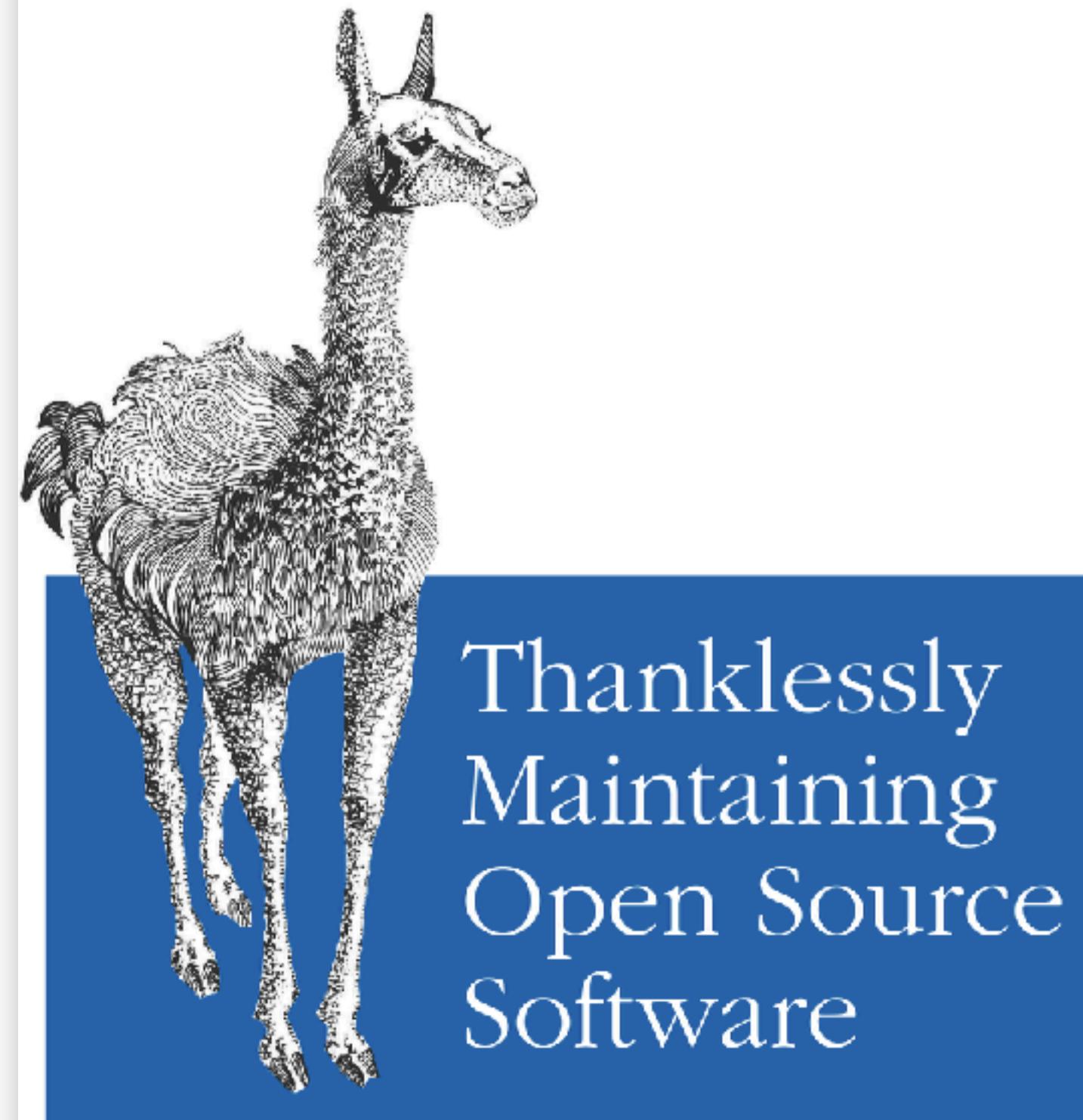
There will be more details to come, but the idea is to have a day where we can work together on anything ranging from submitting your first pull request, to working on your own package. The tidyverse team will be there, so we can help you hit the ground running and/or get over any stumbling blocks that you encounter. Don't have any ideas for something to work on? No problem! We'll be tagging issues in advance to make sure there's lots to do for any- and everyone, regardless of level of expertise.



and a bunch of other places...

giving back

Acting out of the goodness of your heart, or something



@ThePracticalDev

tidyverse team



Works cited

- Traweek, Sharon. 1988. Beamtimes and Lifetimes: The World of High Energy Physics. Cambridge, MA: Harvard University Press.
- NIST image: This work is in the public domain in the United States under the terms of Title 17, Chapter 1, Section 105 of the US Code.
- Pintscher, Lydia, ed. 2012. *Open Advice: FOSS: What We Wish We Had Known When We Started*. <<http://open-advice.org/>>
- "You'll Eventually Know Everything They've Forgotten." In *Open Advice: FOSS: What We Wish We Had Known When We Started*, 29–32.
- Shaikh, Resham. 2018. "Why Women Are Flourishing In R Community But Lagging In Python." <https://reshamas.github.io/why-women-are-flourishing-in-r-community-but-lagging-in-python/>
- "Recommendations to increase the Participation of Women at useR! conferences" R Forwards Taskforce. https://forwards.github.io/docs/recommendations_user/
- Steinmacher, I., Treude, C., & Gerosa, M. A. (2018). Let me in: Guidelines for the Successful Onboarding of Newcomers to Open Source Projects. *IEEE Software*, PP(99), 1. <http://doi.org/10.1109/MS.2018.110162131>
- Steinmacher, I., Gerosa, M., Conte, T. U., & Redmiles, D. F. (2018). Overcoming Social Barriers When Contributing to Open Source Software Projects. *Computer Supported Cooperative Work: CSCW: An International Journal*, 1–44. <http://doi.org/10.1007/s10606-018-9335-z>
- Ford, D., Smith, J., Guo, P. J., & Parnin, C. (2016). Paradise unplugged: identifying barriers for female participation on stack overflow. *Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering - FSE 2016*, 846–857. <http://doi.org/10.1145/2950290.2950331>
- Ford, D., & Parnin, C. (2015). Exploring Causes of Frustration for Software Developers. In *2015 IEEE/ACM 8th International Workshop on Cooperative and Human Aspects of Software Engineering* (pp. 115–116). IEEE. <http://doi.org/10.1109/CHASE.2015.19>
- Casalnuovo, C., Vasilescu, B., Devanbu, P., & Filkov, V. (2015). Developer onboarding in GitHub: the role of prior social links and language experience. *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering - ESEC/FSE 2015*, 817–828. <http://doi.org/10.1145/2786805.2786854>

Thank You

