SUSTAINERS OF THE

tidyverse

Mara Averick (@dataandme)
Tidyverse Developer Advocate, RStudio

Mara Averick

TIDYVERSE DEV ADVOCATE, RSTUDIO

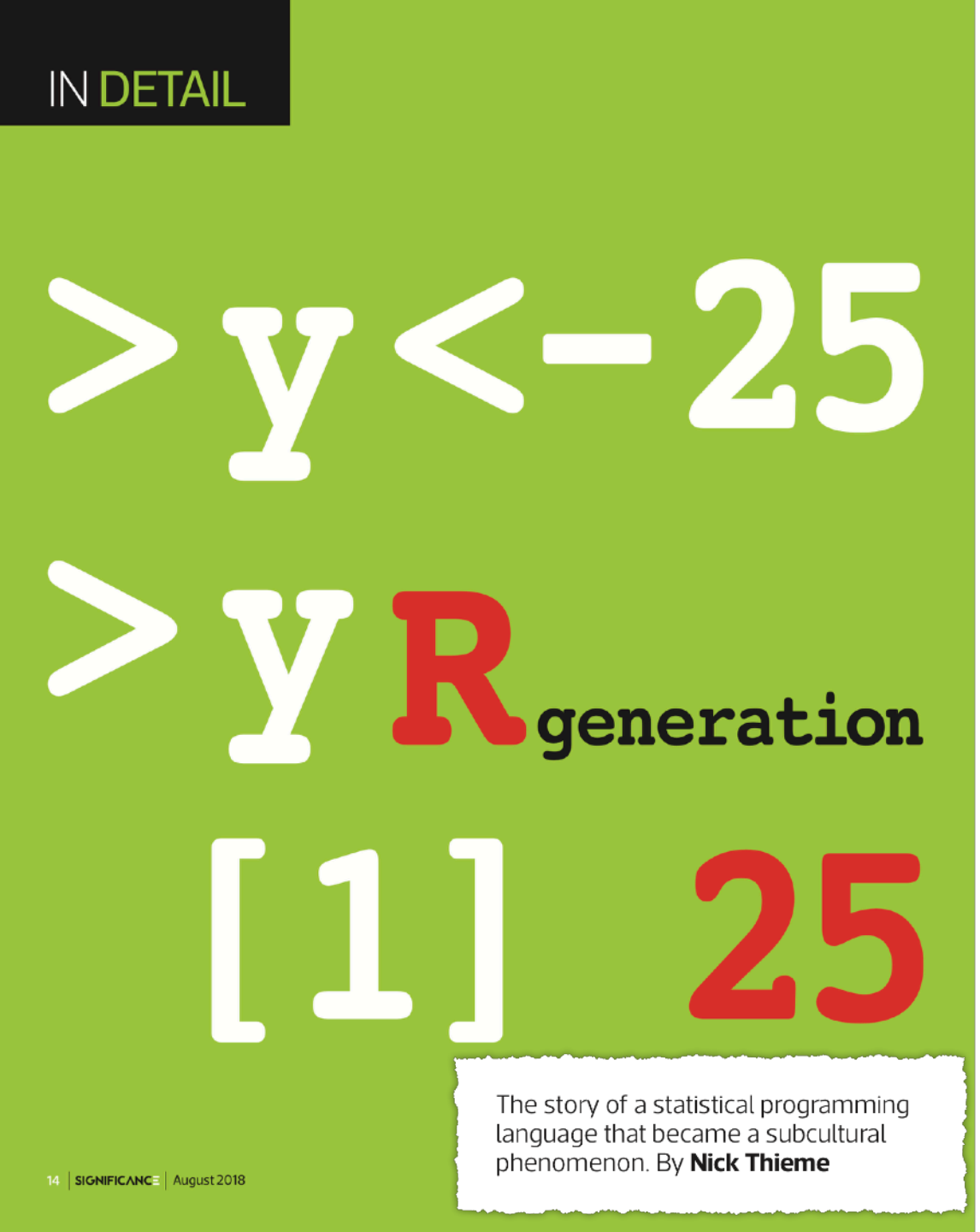
SCIENCE & SOCIETY

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

In Beamtimes and Lifetimes: The World of High Energy Physics. (1988). Cambridge, MA: Harvard University Press.



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 12,000 packages on CRAN

Thieme, N. (2018). R generation. Significance, 15(4), 14-19. http://doi.org/10.1111/j.1740-9713.2018.01169.x

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.

source: https://www.tidyverse.org/

TIDY TOOLS

Functions should be...

SIMPLE

Do one thing and do it well.

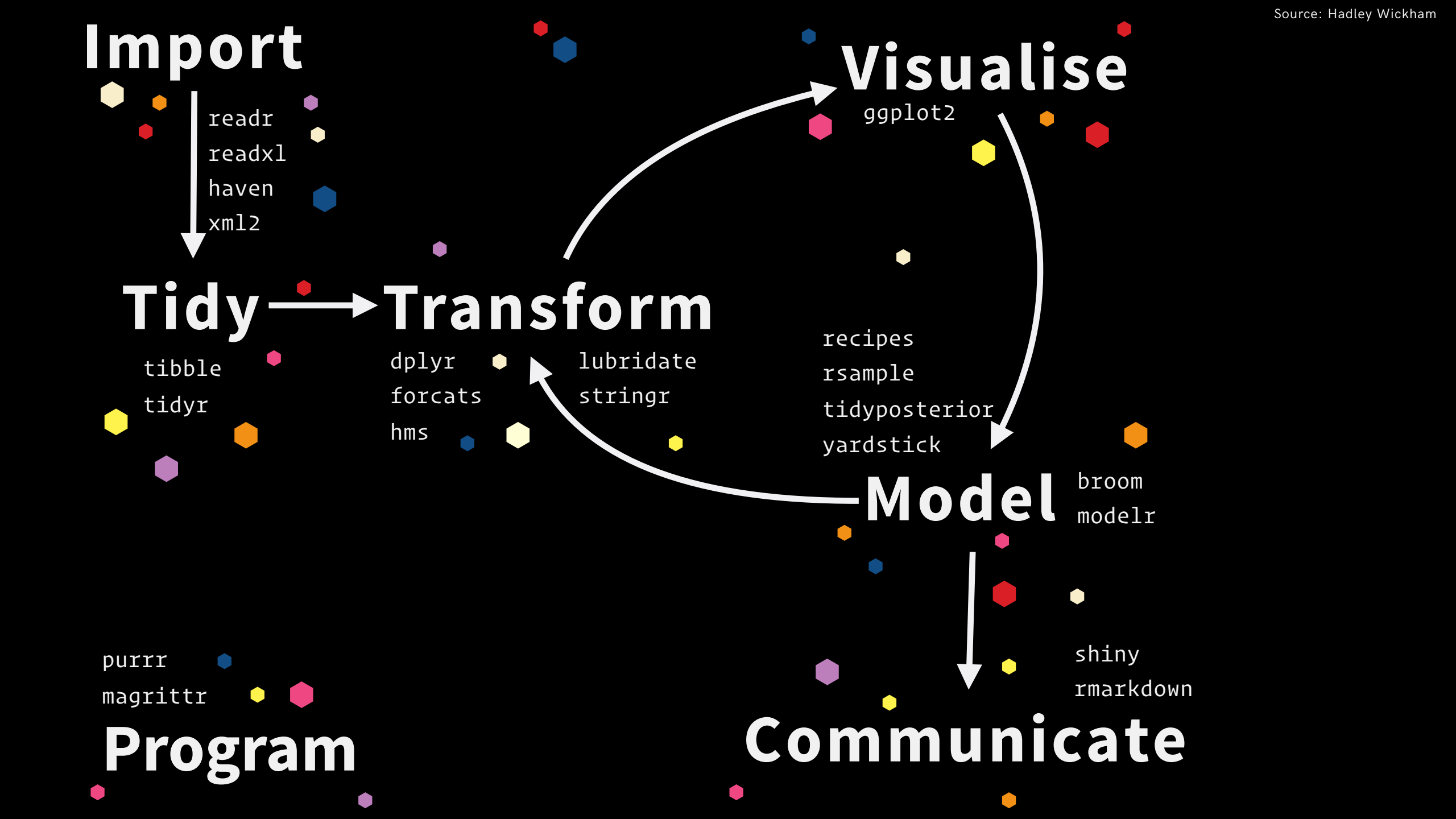
COMPOSABLE

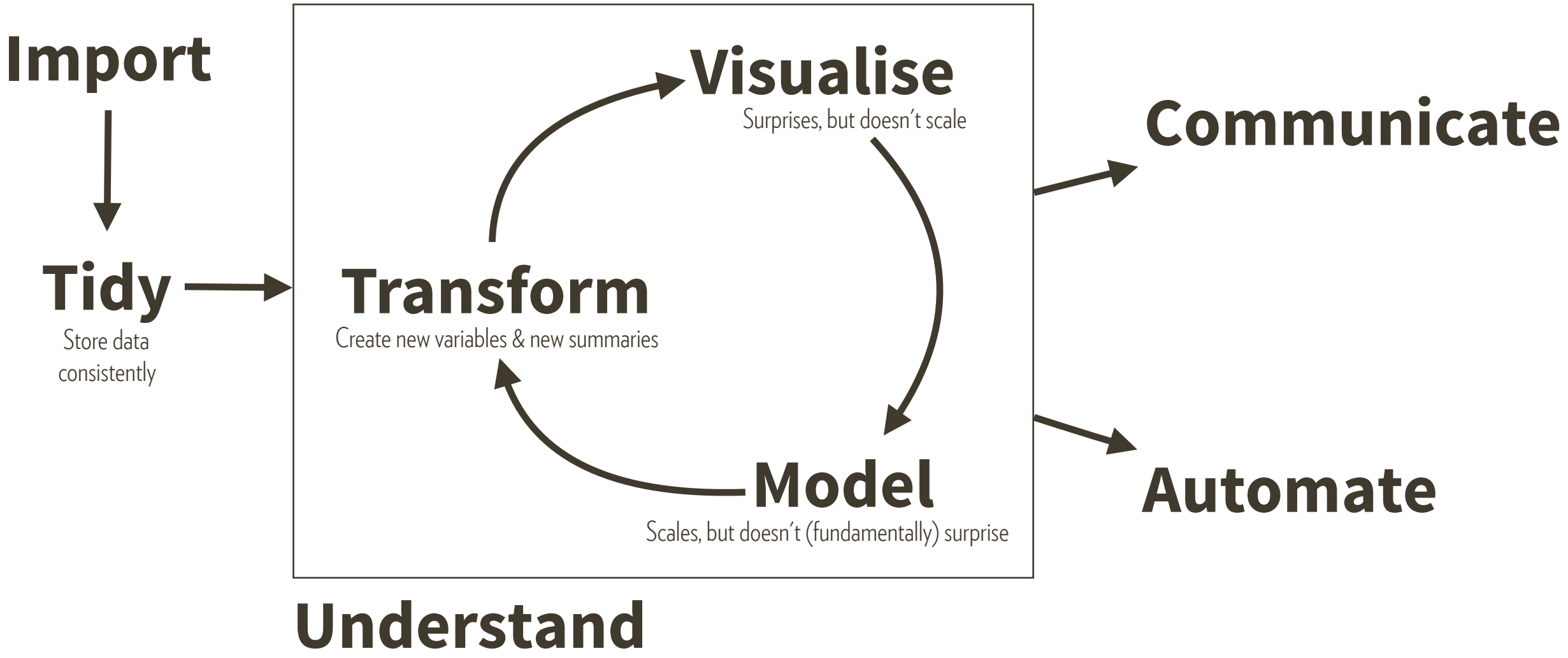
Combine with other functions for multi-step operations.

DESIGNED FOR HUMANS

Use evocative verb names, making them easy to remember.

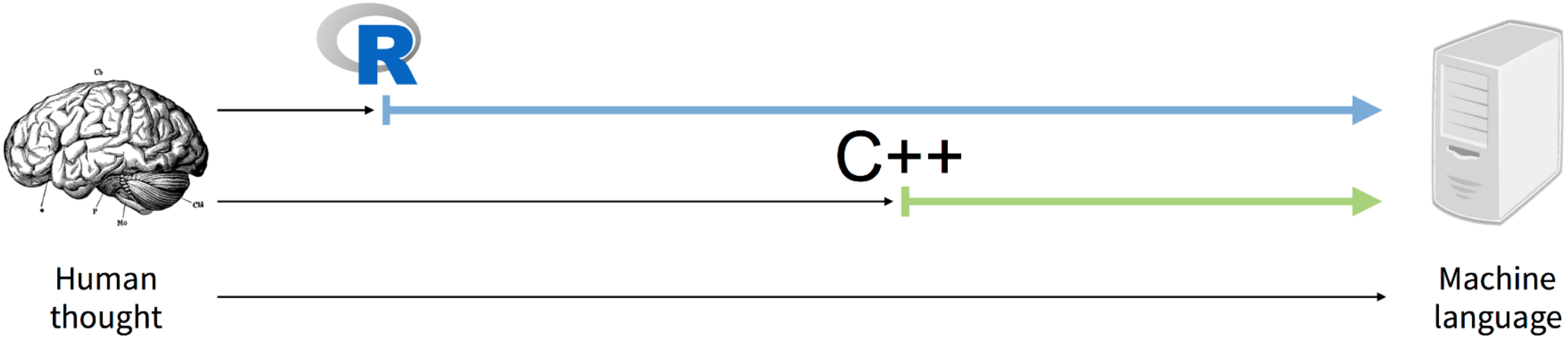
Source: Wickham, Hadley. 2017-11-13. "The tidy tools manifesto." https://cran.r-project.org/web/packages/tidyverse/vignettes/manifesto.html


Source: Hadley Wickham
Import
readr
readxl
haven
xml2
Visualise
ggplot2
Tidy
tibble
tidyr
Transform
dplyr
forcats
hms
lubridate
stringr
recipes
rsample
tidyposterior
yardstick
Model
broom
modelr
purrr
magrittr
Program
shiny
rmarkdown
Communicate

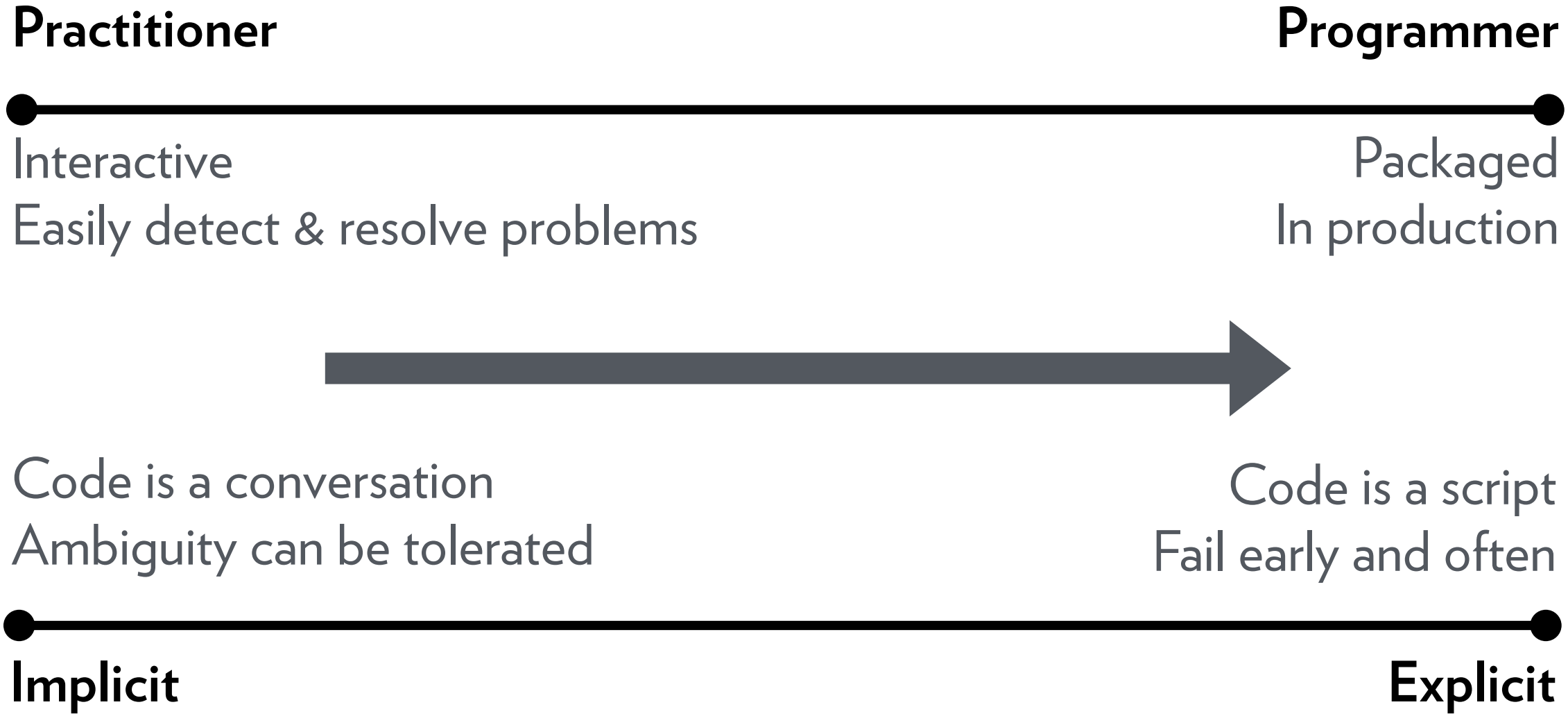


src: Hadley Wickham

A computer language for scientists



via Garrett Grolemund



source: Hadley Wickham

What does this mean for contribution and sustainability?

Studying OSS projects...

"The survival, long-term success, and continuity of [OSS] projects...requires an influx of newcomers... Although many want to volunteer, newcomers' face many difficulties entering OSS; they are akin to explorers finding their way in a hostile landscape."

— Steinmacher, I., Gerosa, M., Conte, T. U., & Redmiles, D. F. (2018). *Overcoming Social Barriers When Contributing to Open Source Software Projects.*

In: Computer Supported Cooperative Work: CSCW: An International Journal, 1-44. http://doi.org/10.1007/s10606-018-9335-z

Key FOSS actors

- maintainers
- contributors
- consumers/users
- sustainers

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

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

THE REPORT

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

Sustain: key recommendations

✓ Create sustainable communities

✓ Free the maintainer

✓ Raise the value of non-code contributions

Report team: Ben Nickolis, Pia Mancini, Justin Dorfman, Robert Gibb <https://sustainoss.org/> 2017

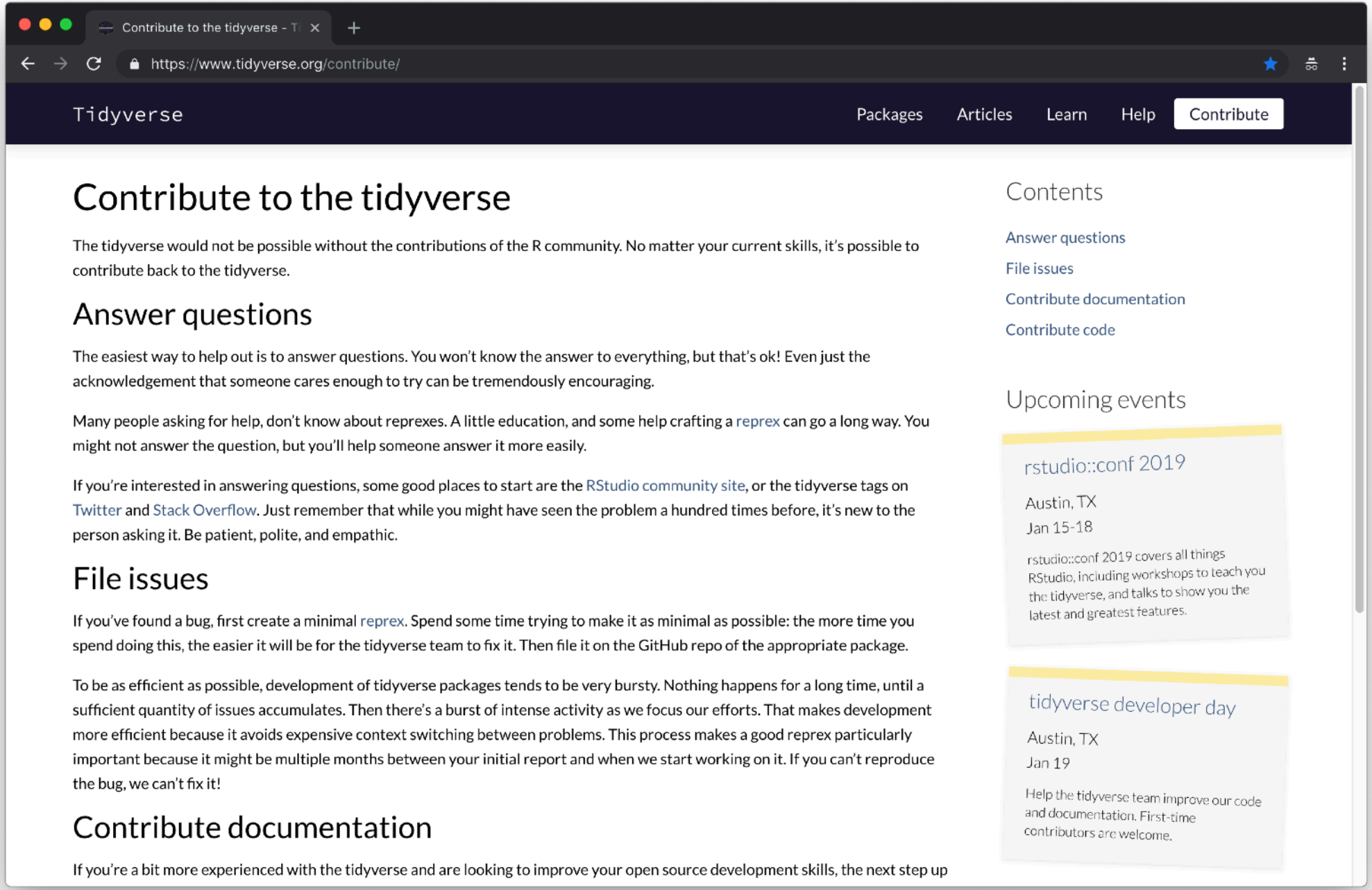
Sustaining the tidyverse…

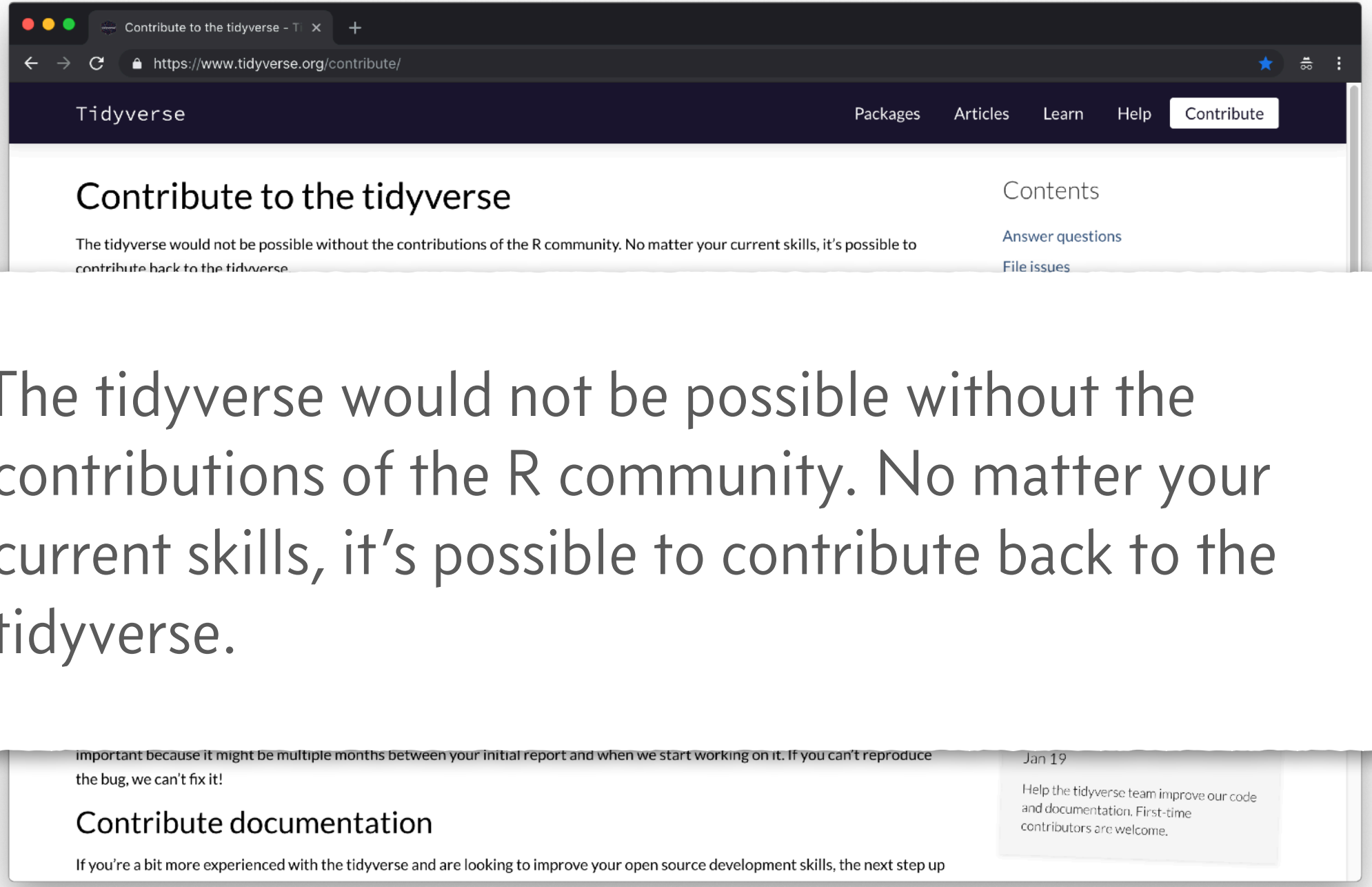
- virtual spaces
- technical tools
- social norms

Sustaining the tidyverse…

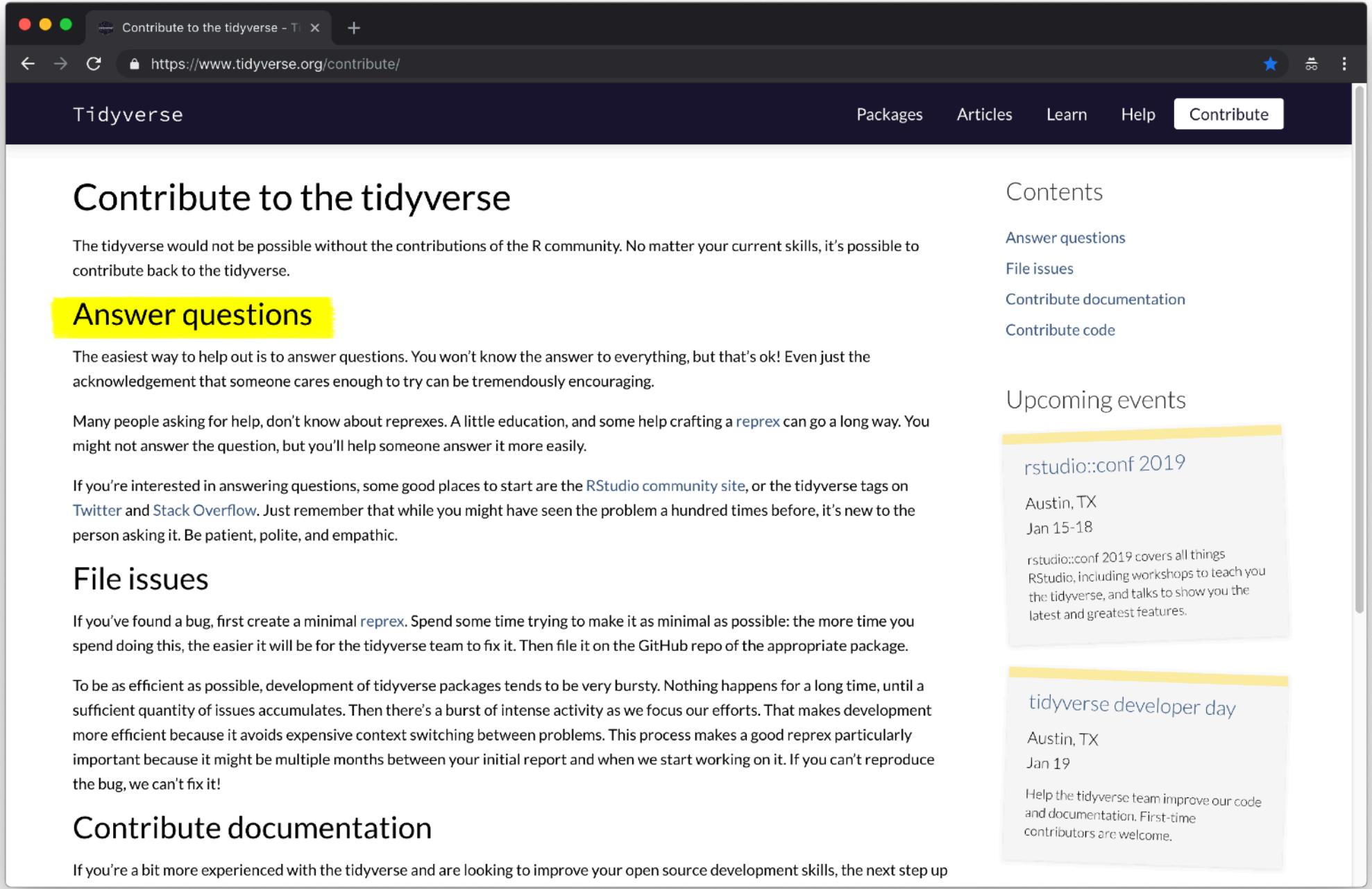
- virtual spaces
- technical tools
- social norms

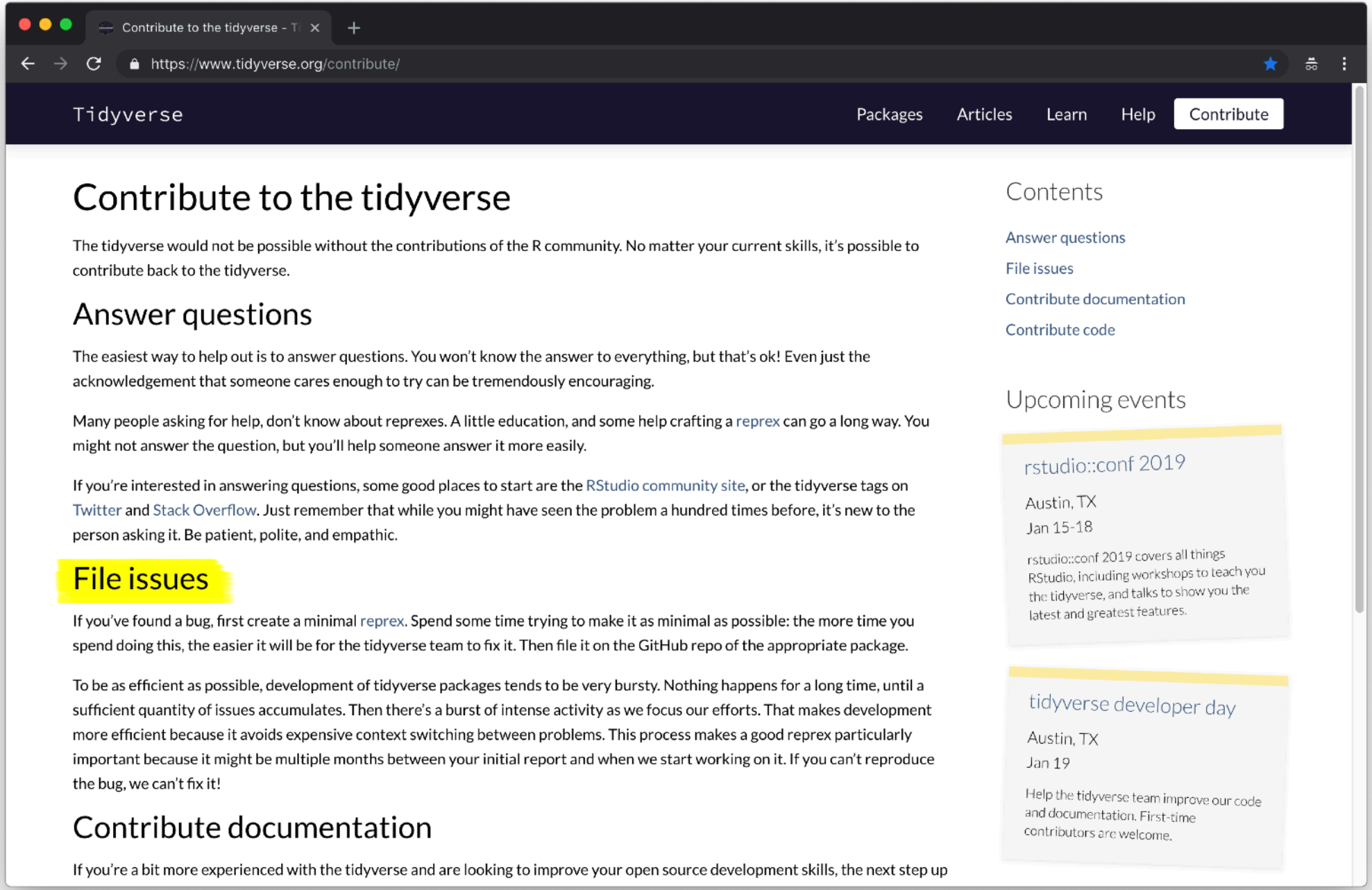
spoiler alert:
they're interrelated


Contribute to the tidyverse - T
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.

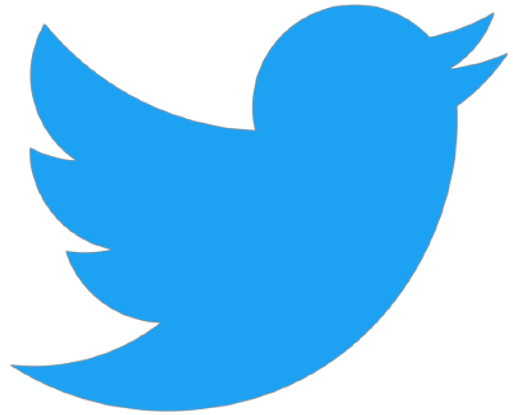


The tidyverse would not be possible without the contributions of the R community. No matter your current skills, it's possible to contribute back to the tidyverse.

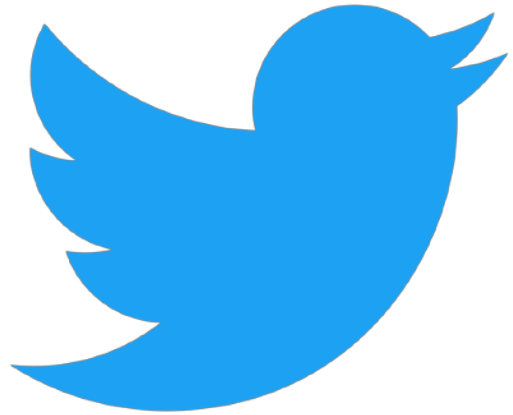

Contribute to the tidyverse - T
https://www.tidyverse.org/contribute/
Tidyverse
Packages
Articles
Learn
Help
Contribute
Contribute to the tidyverse
The tidyverse would not be possible without the contributions of the R community. No matter your current skills, it's possible to contribute back to the 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 - T
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?




ROpenSci



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
- 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 Ford[1], **Kristina Lustig**[2], **Jeremy Banks**[2], **Chris Parnin**[1]
[1]North Carolina State University, Raleigh, NC, USA
[2]Stack 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.

CHI 2018, April 21–26, 2018, Montreal, QC, Canada
© 2018 ACM. ISBN 978-1-4503-5620-6/18/04…$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, 33] 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 [33] 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

Ford, Denae, Kristina Lustig, Jeremy Banks, Chris Parnin, and North Carolina. 2018. "'We Don't Do That Here': How Collaborative Editing with Mentors Improves Engagement in Social Q & A Communities." In CHI 2018. Montreal, QC, Canada: ACM. doi:10.1145/3173574.3174182.

"We don't do that here"

- Question phrasing
- Formatting posts format code as code
- 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 Ford[1], Kristina Lustig[2], Jeremy Banks[2], Chris Parnin[1]
[1]North Carolina State University, Raleigh, NC, USA
[2]Stack 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.

CHI 2018, April 21–26, 2018, Montreal, QC, Canada
© 2018 ACM. ISBN 978-1-4503-5620-6/18/04…$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, 33] 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 [33] 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

Ford, Denae, Kristina Lustig, Jeremy Banks, Chris Parnin, and North Carolina. 2018. "'We Don't Do That Here': How Collaborative Editing with Mentors Improves Engagement in Social Q & A Communities." In CHI 2018. Montreal, QC, Canada: ACM. doi:10.1145/3173574.3174182.

"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 Ford[1], Kristina Lustig[2], Jeremy Banks[2], Chris Parnin[1]
[1]North Carolina State University, Raleigh, NC, USA
[2]Stack 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.

CHI 2018, April 21–26, 2018, Montreal, QC, Canada
© 2018 ACM. ISBN 978-1-4503-5620-6/18/04…$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, 33] 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 [33] 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

Ford, Denae, Kristina Lustig, Jeremy Banks, Chris Parnin, and North Carolina. 2018. "'We Don't Do That Here': How Collaborative Editing with Mentors Improves Engagement in Social Q & A Communities." In CHI 2018. Montreal, QC, Canada: ACM. doi:10.1145/3173574.3174182.

"We don't do that here"

- Question phrasing
- Formatting posts
- Community triage
- Question framing clarity, research of problem, context
- Community culture of asking

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

Denae Ford[1], Kristina Lustig[2], Jeremy Banks[2], Chris Parnin[1]
[1]North Carolina State University, Raleigh, NC, USA
[2]Stack 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.

CHI 2018, April 21–26, 2018, Montreal, QC, Canada
© 2018 ACM. ISBN 978-1-4503-5620-6/18/04…$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, 33] 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 [33] 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

Ford, Denae, Kristina Lustig, Jeremy Banks, Chris Parnin, and North Carolina. 2018. "'We Don't Do That Here': How Collaborative Editing with Mentors Improves Engagement in Social Q & A Communities." In CHI 2018. Montreal, QC, Canada: ACM. doi:10.1145/3173574.3174182.

"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 Ford[1], Kristina Lustig[2], Jeremy Banks[2], Chris Parnin[1]
[1]North Carolina State University, Raleigh, NC, USA
[2]Stack 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.

CHI 2018, April 21–26, 2018, Montreal, QC, Canada
© 2018 ACM. ISBN 978-1-4503-5620-6/18/04…$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, 33] 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 [33] 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.

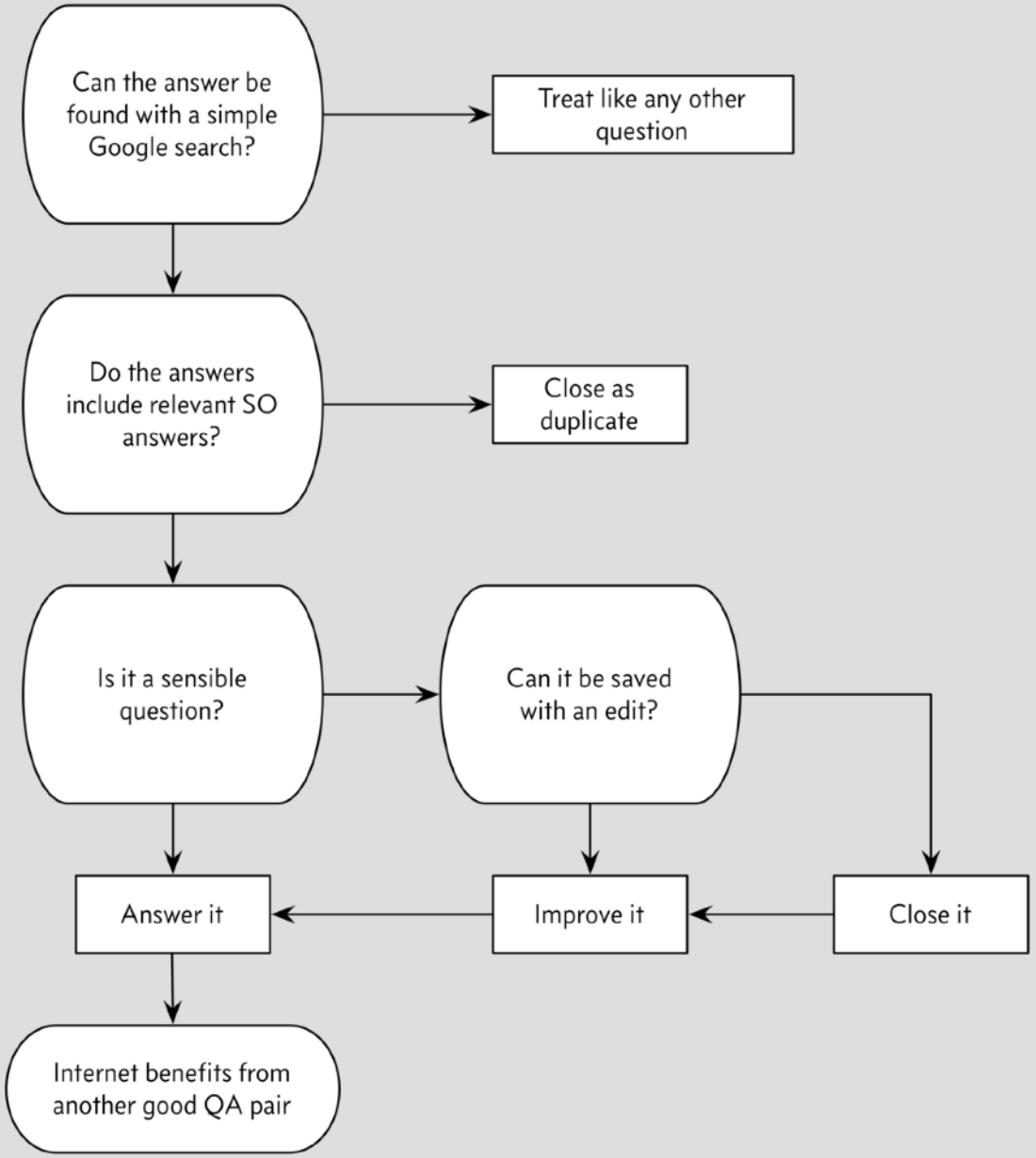
In this paper, we applied theory related to learning and communities of practice to a social Q&A site, by using methods related to mutual engagement and formative feedback to improve novices' questions. Building on design claims for increasing engagement in online communities [19], we created *Help Rooms* 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

Ford, Denae, Kristina Lustig, Jeremy Banks, Chris Parnin, and North Carolina. 2018. "'We Don't Do That Here': How Collaborative Editing with Mentors Improves Engagement in Social Q & A Communities." In CHI 2018. Montreal, QC, Canada: ACM. doi:10.1145/3173574.3174182.

How should you respond to RTFM questions?

goal = good QA pair



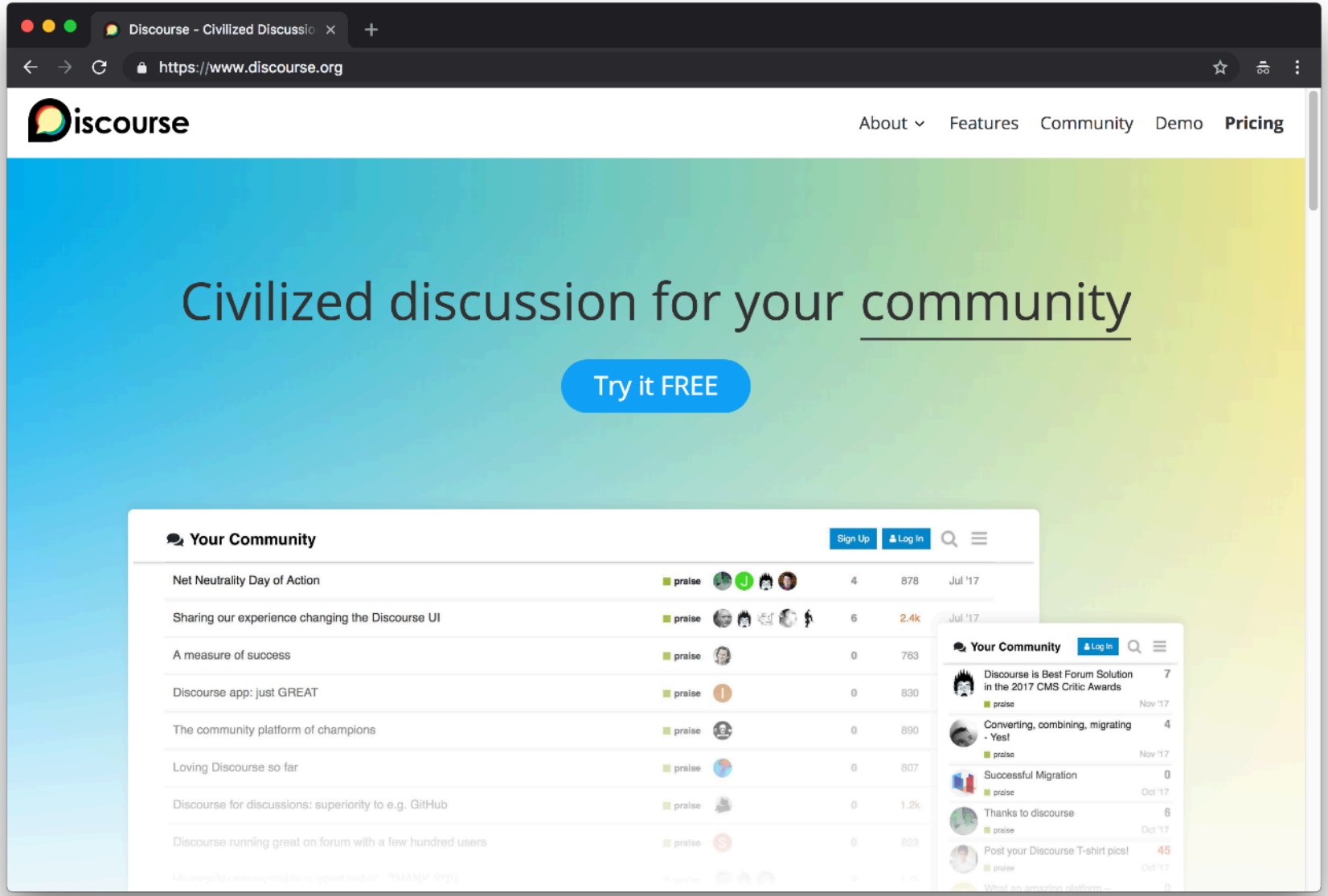
Source: Shawn Chin https://meta.stackexchange.com/a/161346/213022

10 simple rules for getting help from online scientific communities

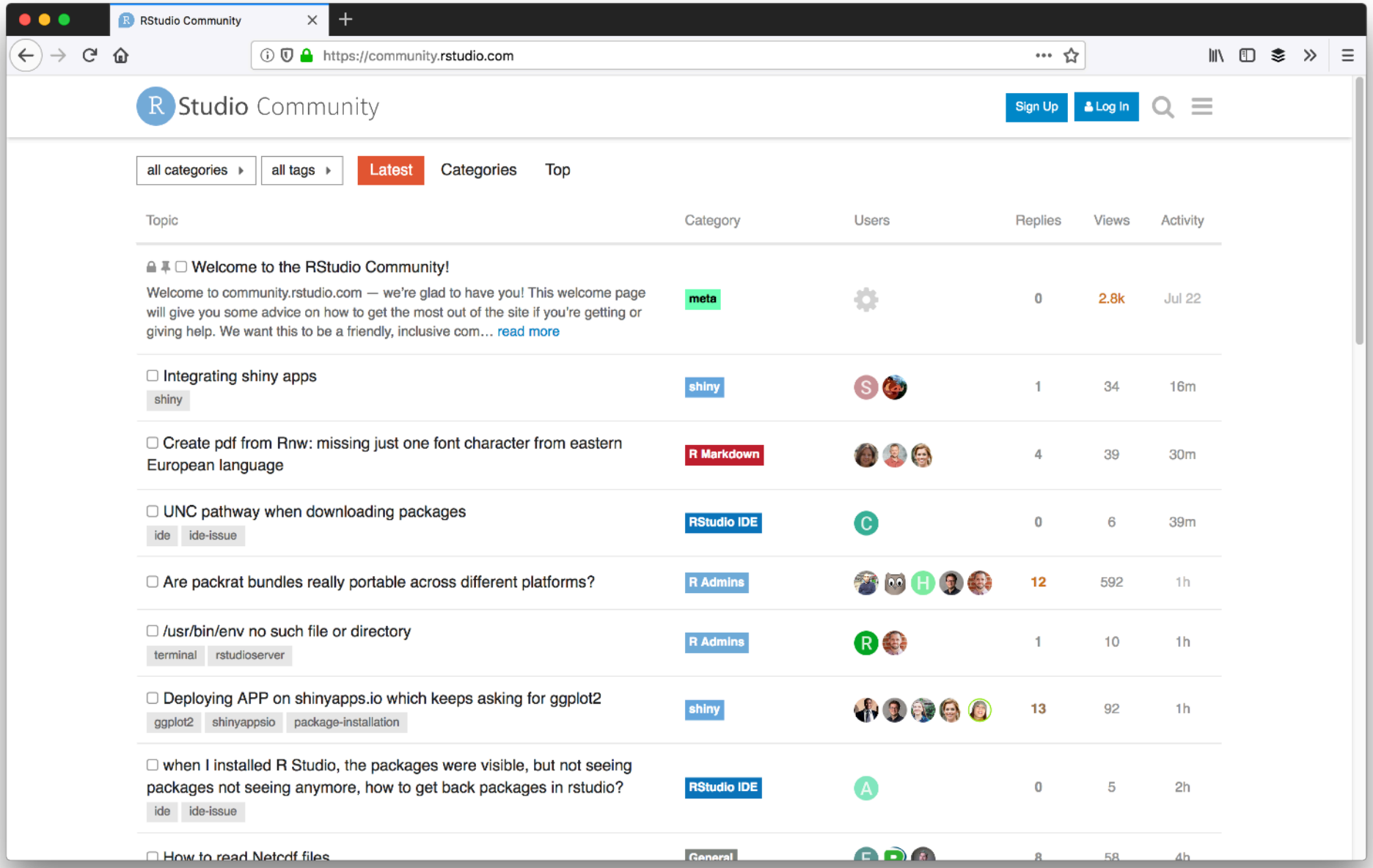
1. Do not be afraid to ask a question
2. State the question clearly
3. New to a mailing list? Learn the established customs before posting
4. Do not ask what has already been answered
5. Always use a good title
6. Do your homework before posting
7. Proofread your post and write in correct English
8. 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

Dall'Olio, Giovanni M., Jacopo Marino, Michael Schubert, Kevin L. Keys, Melanie I. Stefan, Colin S. Gillespie, Pierre Poulain, et al. 2011. *"Ten Simple Rules for Getting Help from Online Scientific Communities." PLoS Computational Biology 7 (9): 10–12. doi:10.1371/journal.pcbi.1002202.*

RStudio Community



https://www.discourse.org/

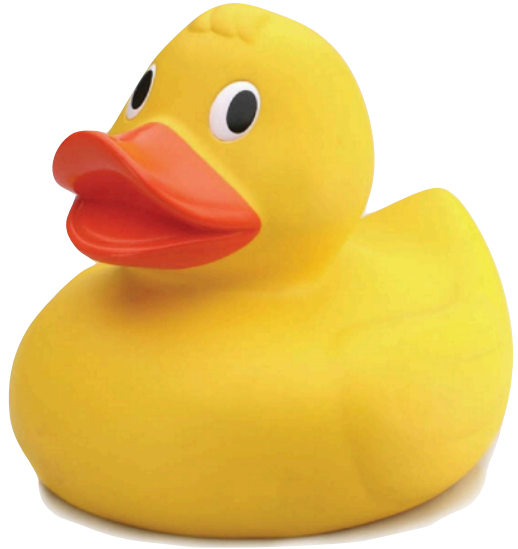


https://community.rstudio.com

The art of the question

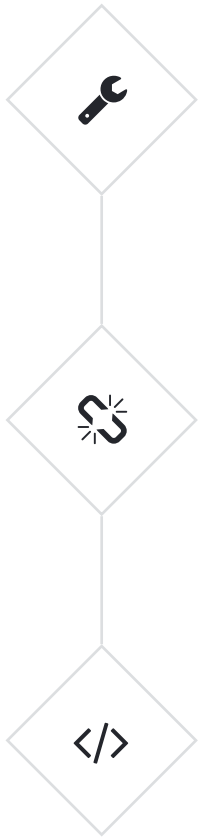
The most useless problem statement that one can face is "it doesn't work", yet we seem to get it far too often.

– Thiago Maciera



Maciera, Thiago. 2012. "The Art of Problem Solving." In Open Advice: FOSS: What We Wish We Had Known When We Started, edited by Lydia Pintscher, 55-61.

the anatomy of an issue

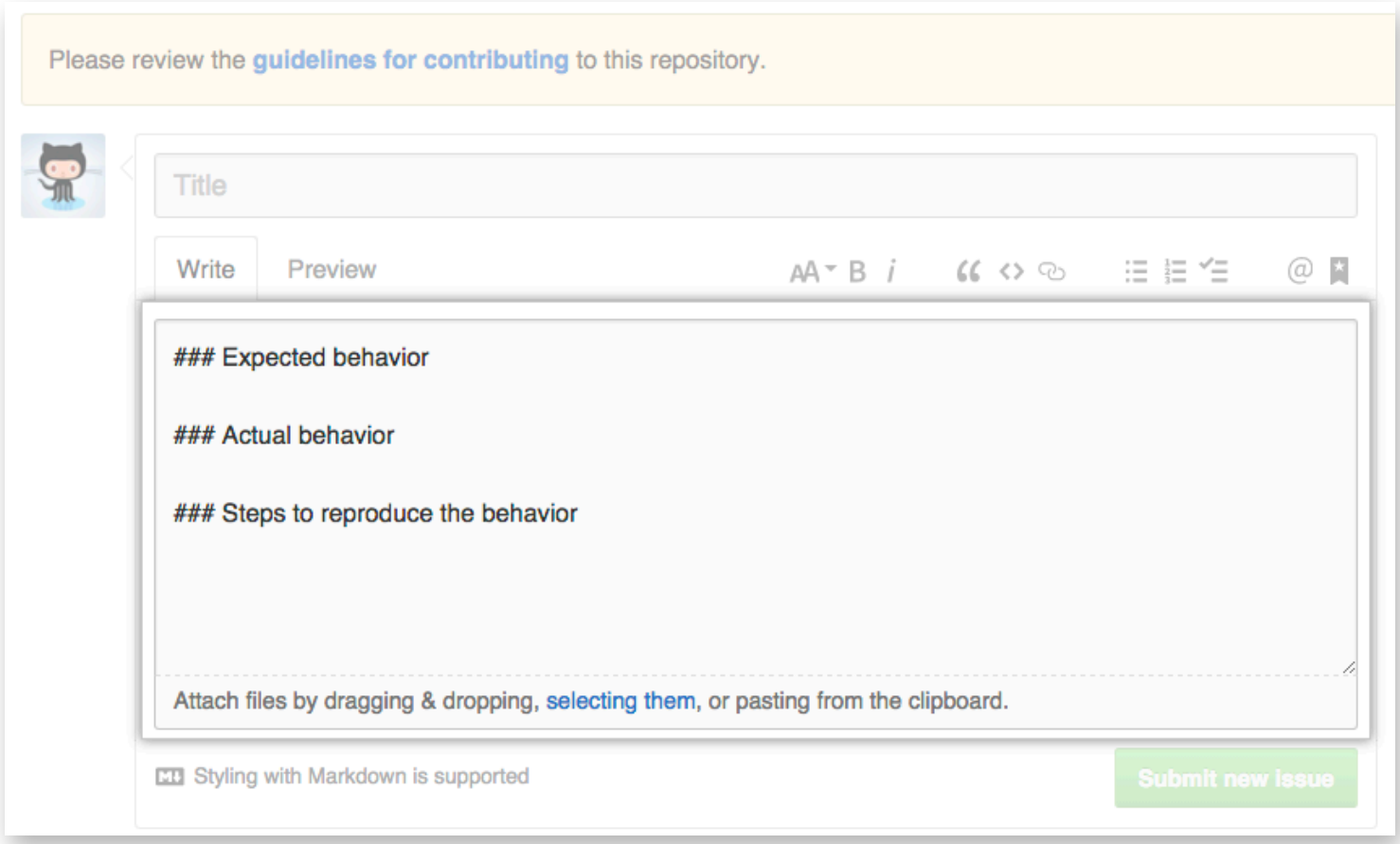


PROBLEM DESCRIPTION

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 -

Steps ta

Steps taken so far - i tried and tried..

System

- P
- R
- C
- R Version:

Also:

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

From Troubleshooting Guide: Using RStudio

Error Information: Packages missing

Description of issue -

Steps ta

Steps taken so far - i tried and tried..

System

- P
- R
- C
- R Version:

Also:

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

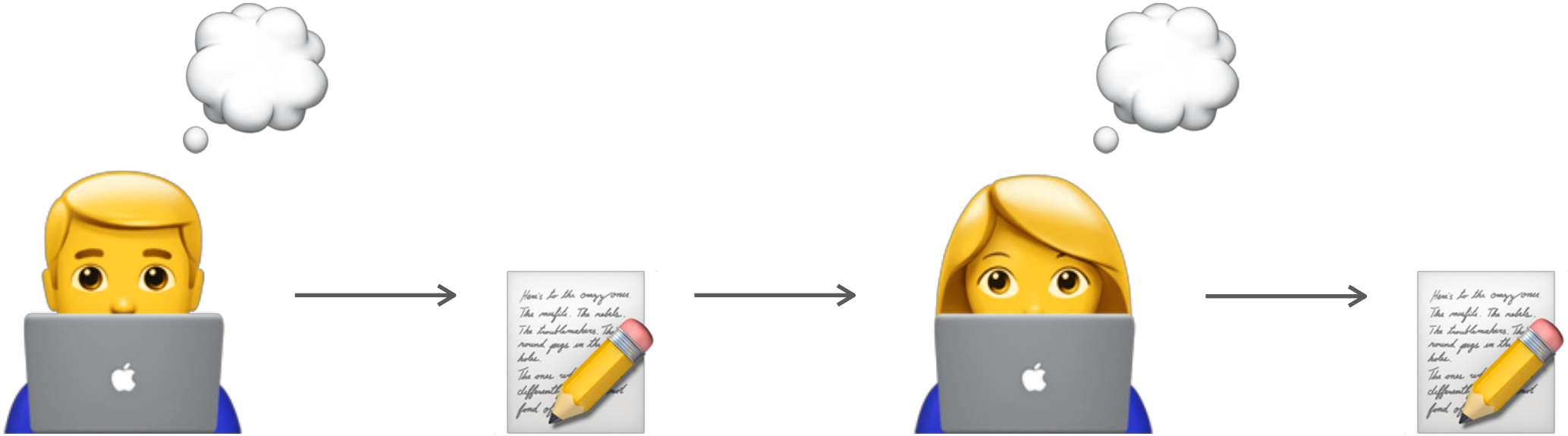
From Troubleshooting Guide: Using RStudio

They're not wrong

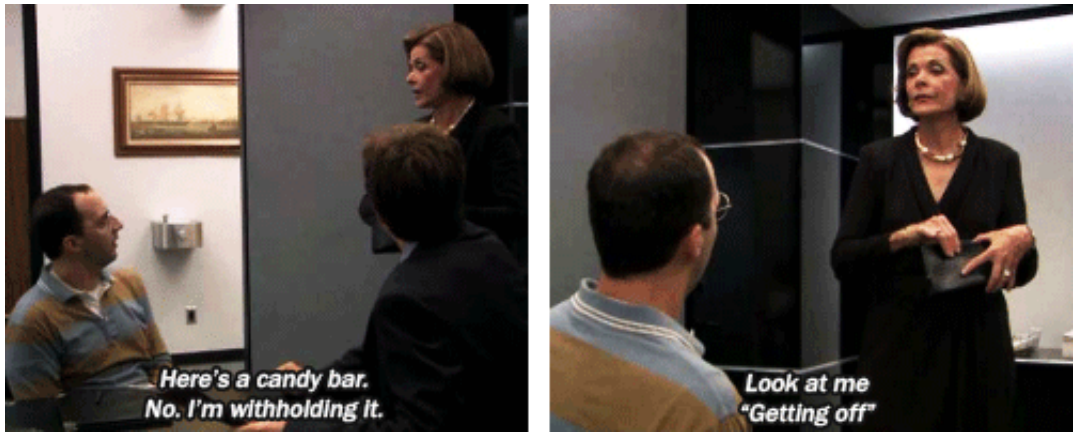
“It is impossible to speak in such a way that you cannot be misunderstood.”

— Karl Popper

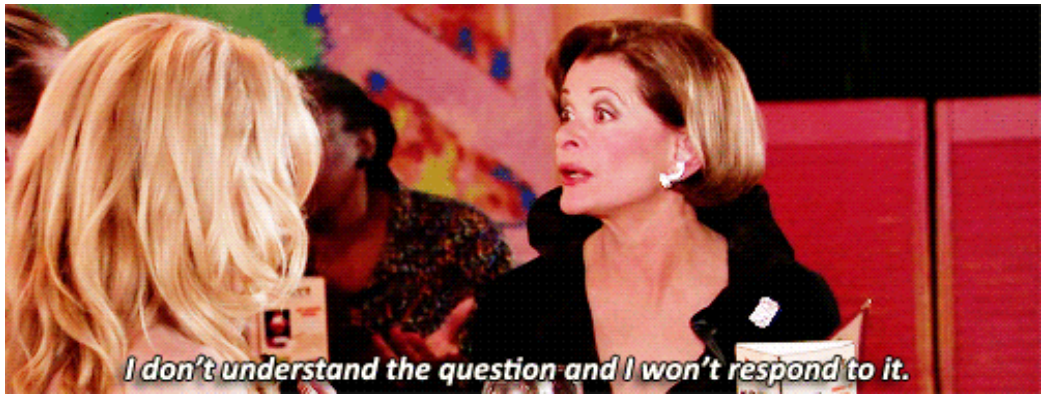
Writing prose about code...



Writing prose about code...



Writing prose about code...



the magic of reprex

reproducible
example

reprex raison d'être

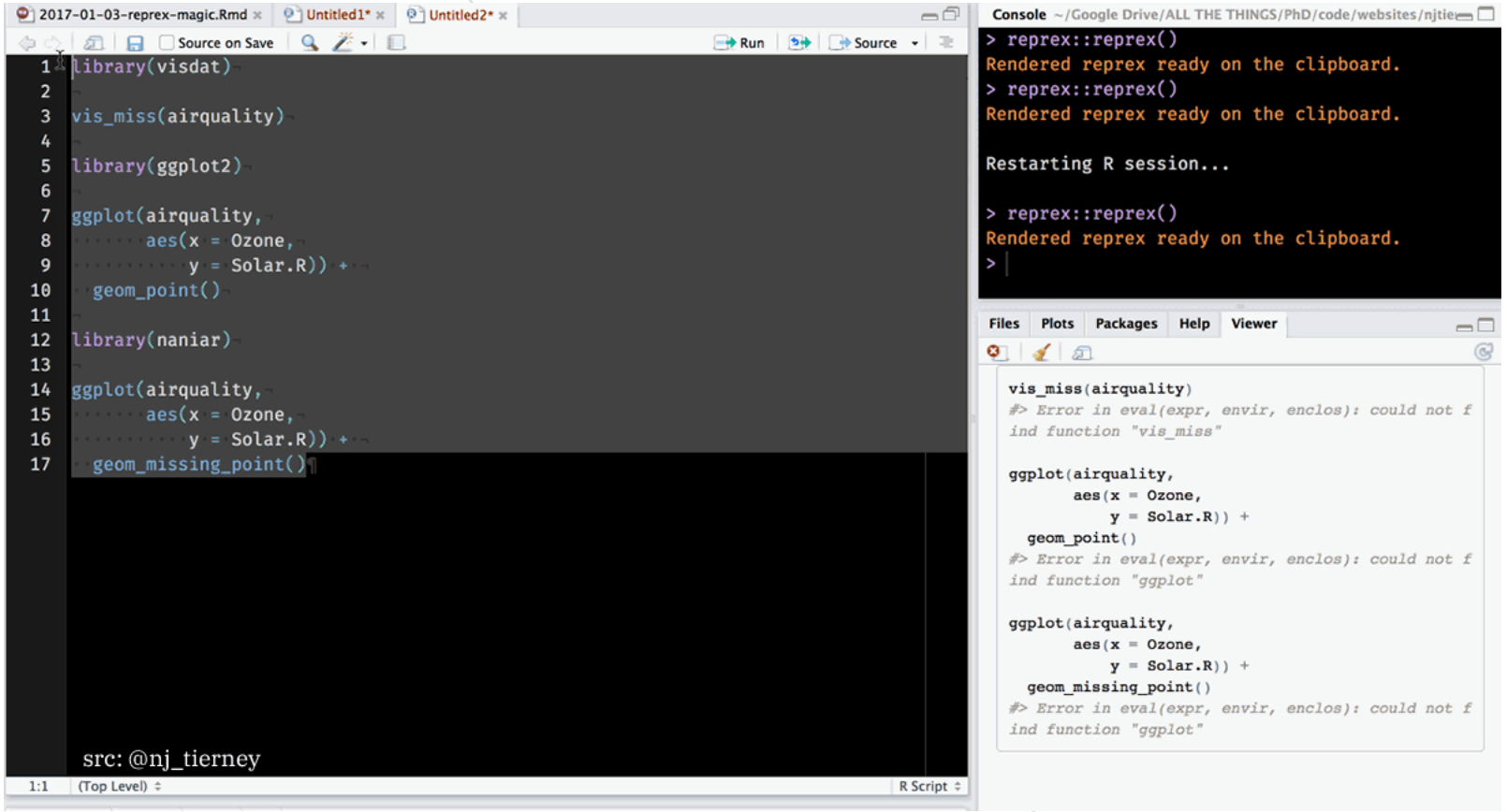
Keys to reprex-cellence

✓ Code that **actually runs**

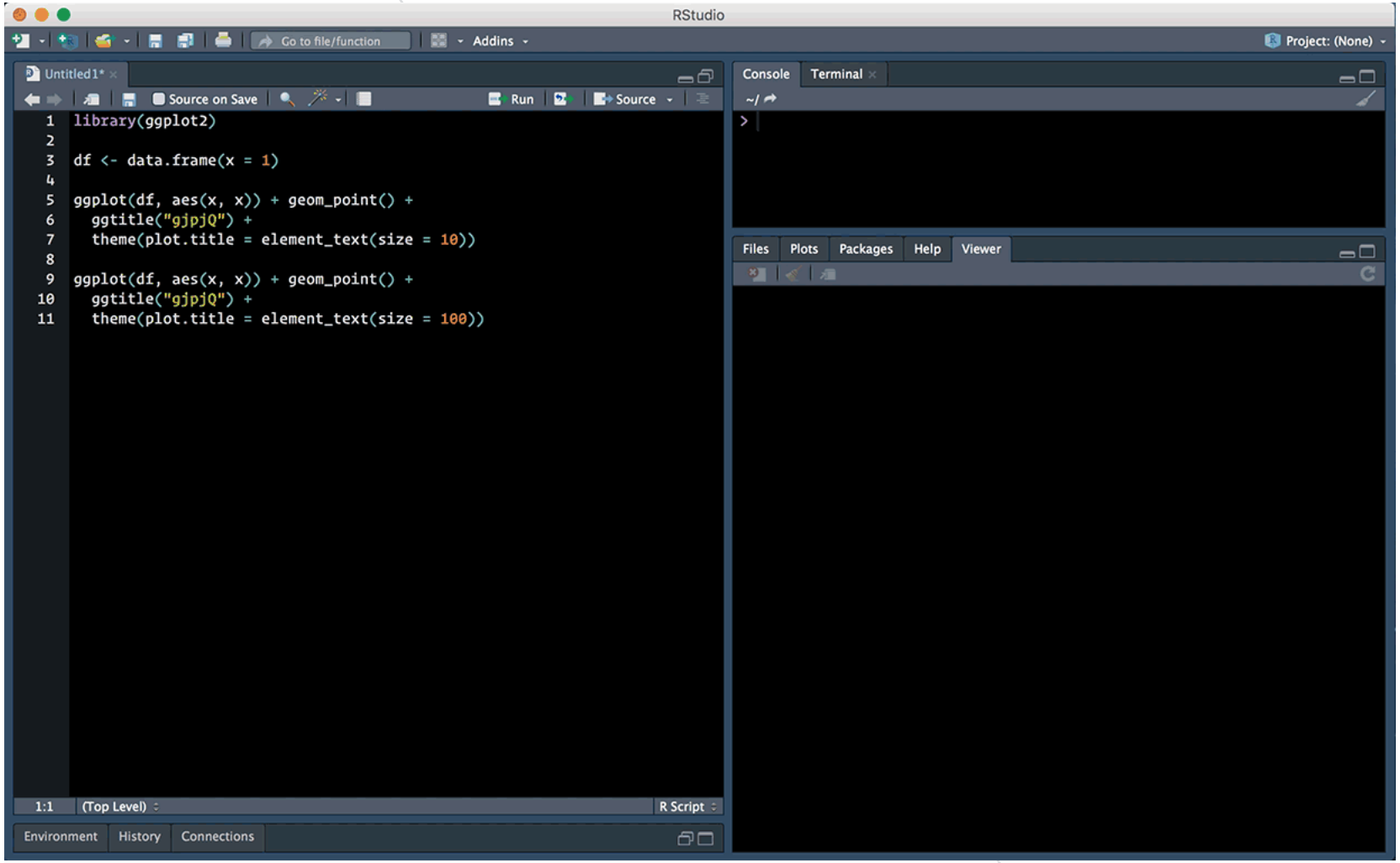
✓ Code that **doesn't have to be run**

✓ Code that **can be easily run**

Source: Jenny Bryan, 2017. "reprex: the package, the point." https://speakerdeck.com/jennybc/reprex-help-me-help-you



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



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

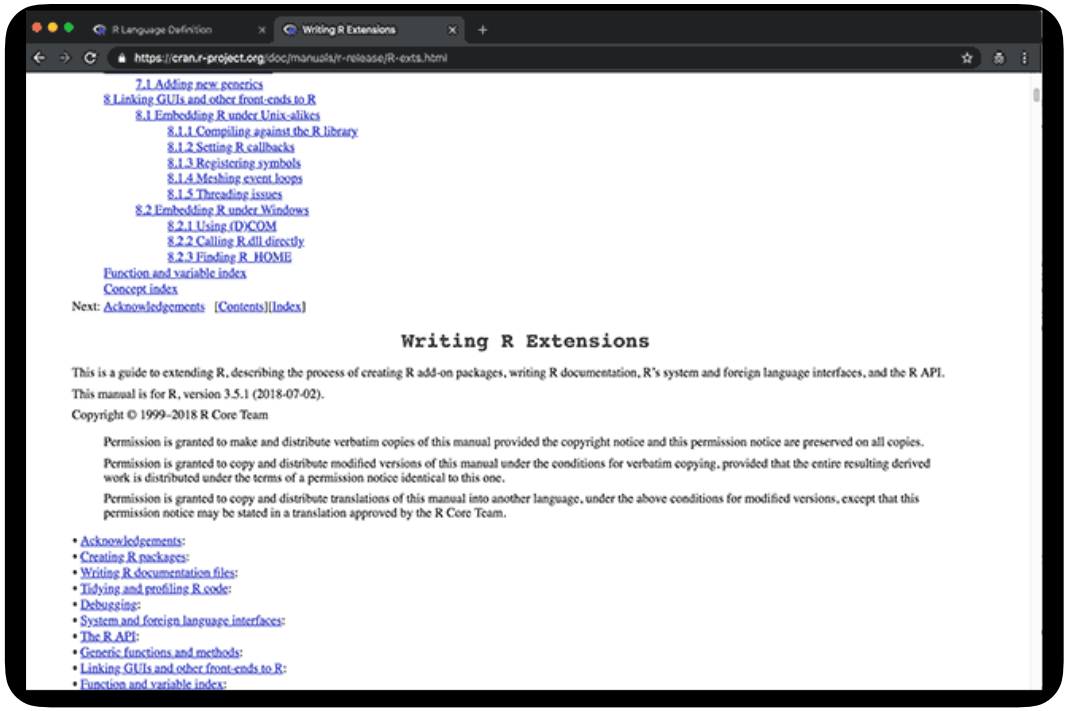
Resources, videos, we've got it all...

RTFM

The R Core Team "Writing R Extensions." R, version 3.5.1 (2018-07-02). https://cran.r-project.org/doc/manuals/r-release/R-exts.html Copyright © 1999-2018 R Core Team

RTFM

TFM →



The R Core Team "Writing R Extensions." R, version 3.5.1 (2018-07-02). https://cran.r-project.org/doc/manuals/r-release/R-exts.html Copyright © 1999-2018 R Core Team

Guide them through the docs

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

tidyverse tidyr regex

mara Sustainer May 20

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.

RegExr.com 21 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/ 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 occurence of character

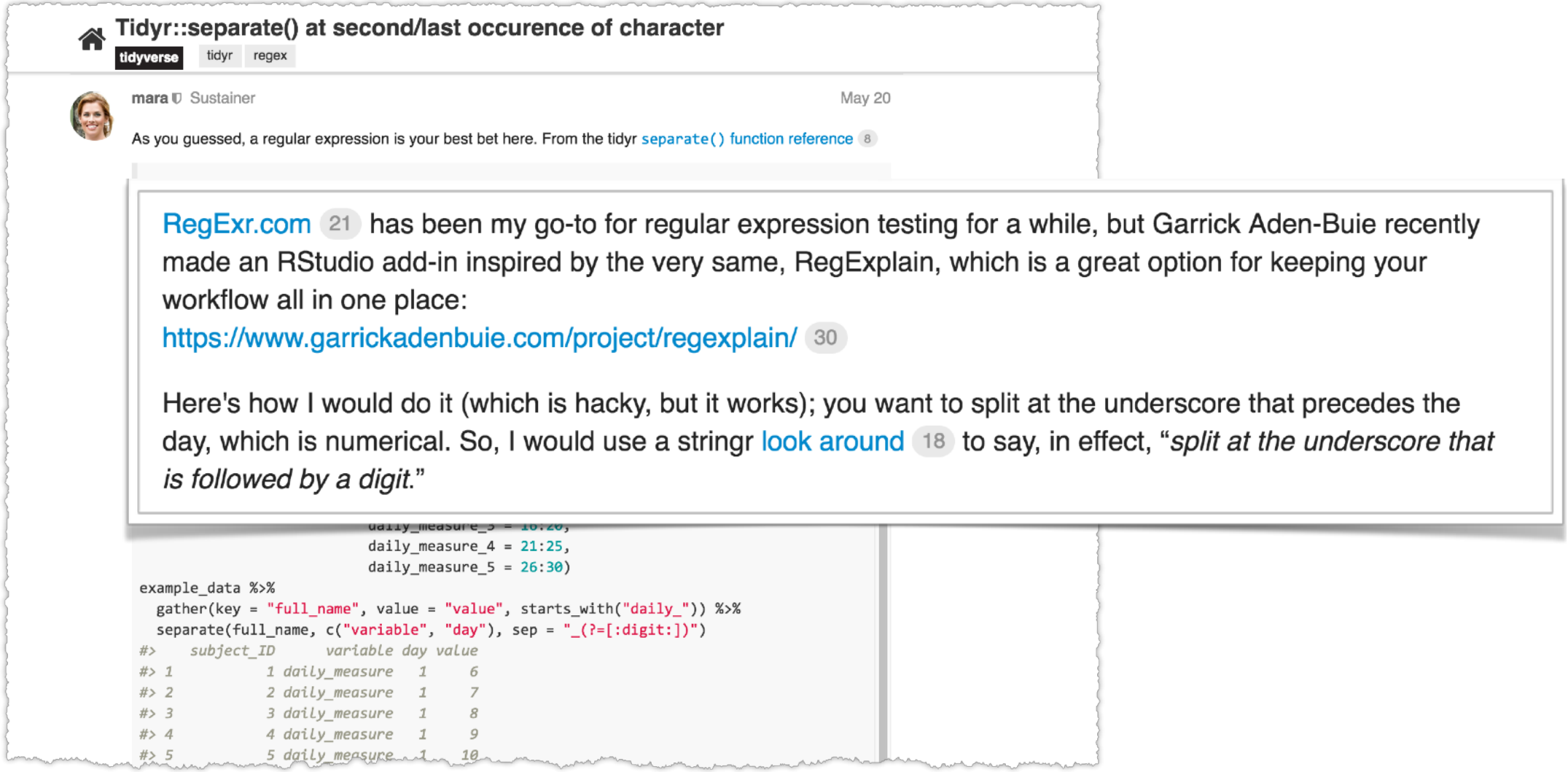
> As you guessed, a regular expression is your best bet here. From the tidyr `separate()` function reference 8
>
> > Given either regular expression or a vector of character positions, `separate()` turns a single character column into multiple columns.

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 occurence of character

tidyverse tidyr regex

mara Sustainer May 20

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

RegExr.com 21 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/ 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*."

```
                          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
```

Contributing to FOSS

WHAT HOLDS PEOPLE BACK?

Sail Fish Scuba https://sailfishscuba.com/manowar/

Pintscher, Lydia, Ed. 2012. *Open Advice: Foss: What We Wish We Had Known When We Started.*

Contributing to FOSS

WHAT HOLDS PEOPLE BACK?

- *“I can't write code.”*

Sail Fish Scuba https://sailfishscuba.com/manowar/

Pintscher, Lydia, Ed. 2012. *Open Advice: Foss: What We Wish We Had Known When We Started.*

Contributing to FOSS

WHAT HOLDS PEOPLE BACK?

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

Sail Fish Scuba https://sailfishscuba.com/manowar/

Pintscher, Lydia, Ed. 2012. *Open Advice: Foss: What We Wish We Had Known When We Started.*

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.”*

Sail Fish Scuba https://sailfishscuba.com/manowar/

Pintscher, Lydia, Ed. 2012. *Open Advice: Foss: What We Wish We Had Known When We Started.*

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.”*

Sail Fish Scuba https://sailfishscuba.com/manowar/

Pintscher, Lydia, Ed. 2012. *Open Advice: Foss: What We Wish We Had Known When We Started.*

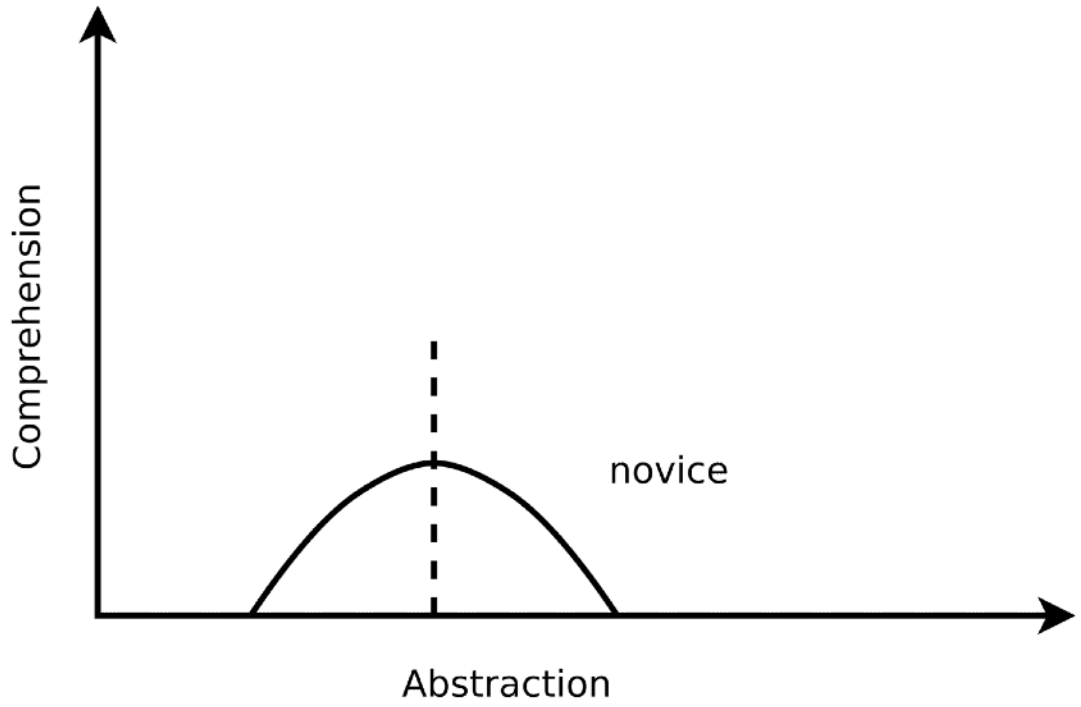
The newcomer's paradox...

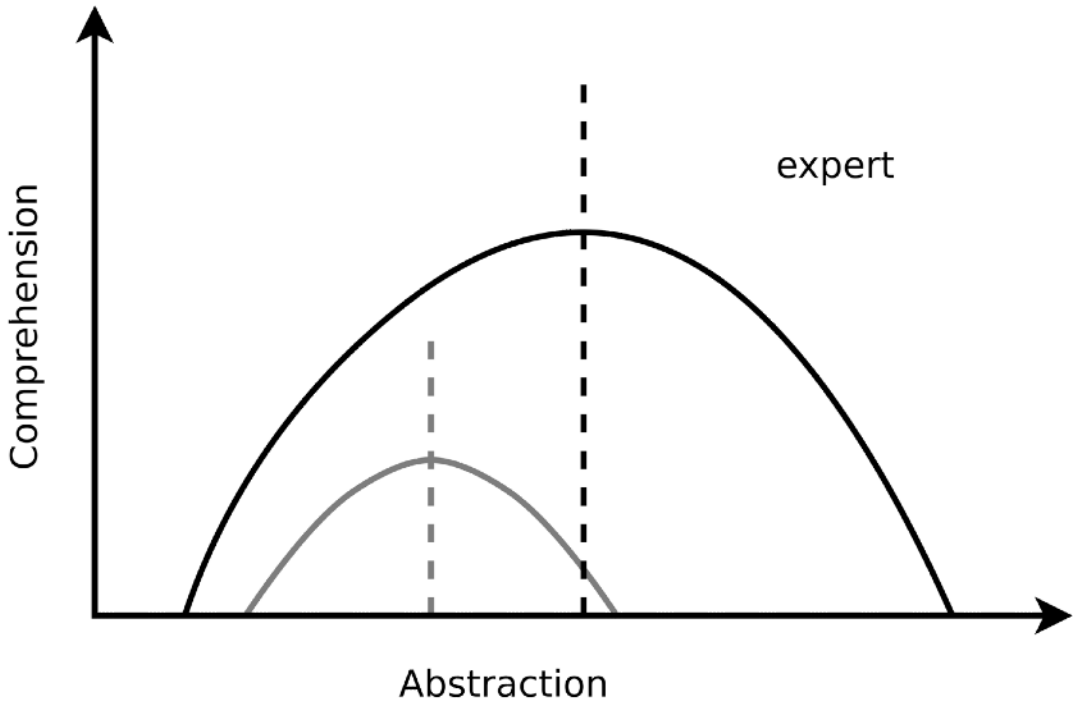
When you ask for help, some friendly soul will no doubt tell you that "it's easy, just do foo, bar and baz." *Except for you, it is not easy, there may be no documentation for foo, bar is not doing what it is supposed to be doing and what is this baz thing anyway with its eight disambiguation entries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

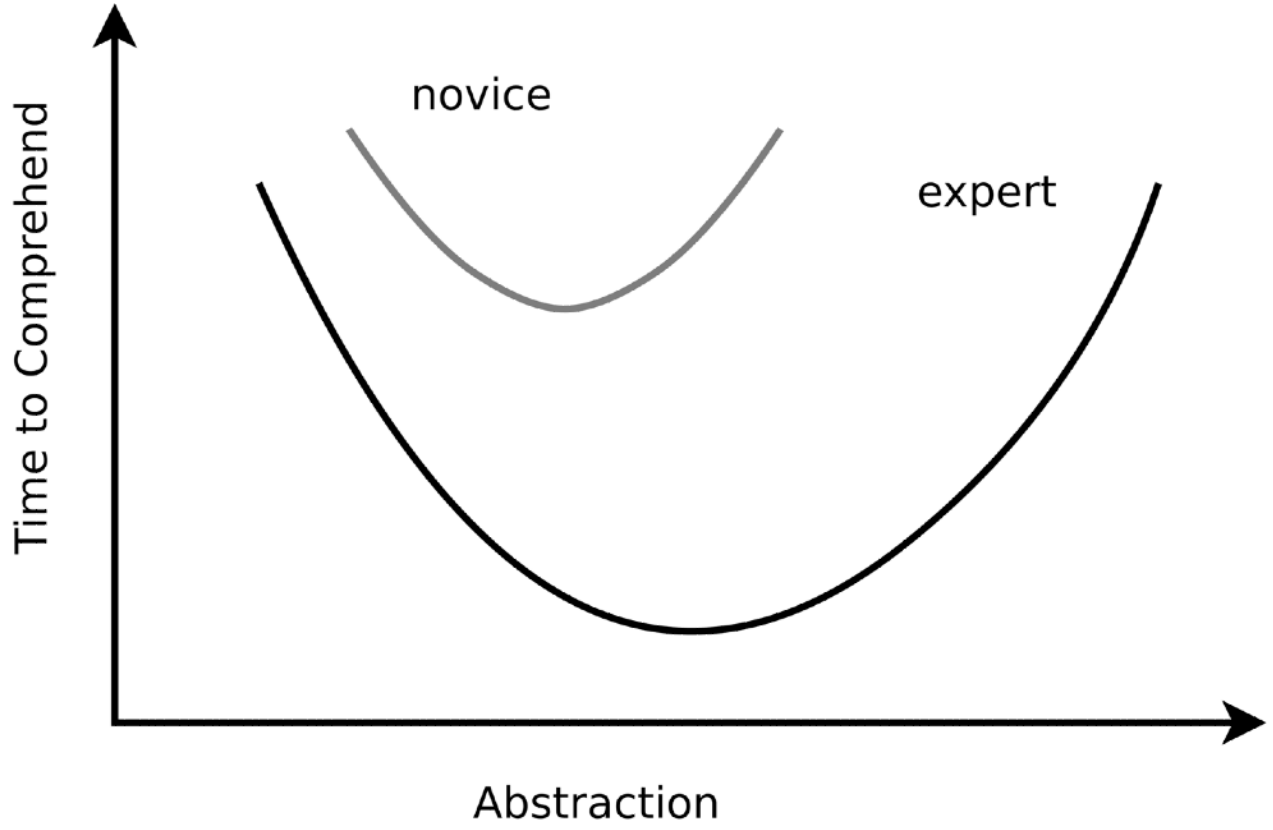
The comprehension comfort curve





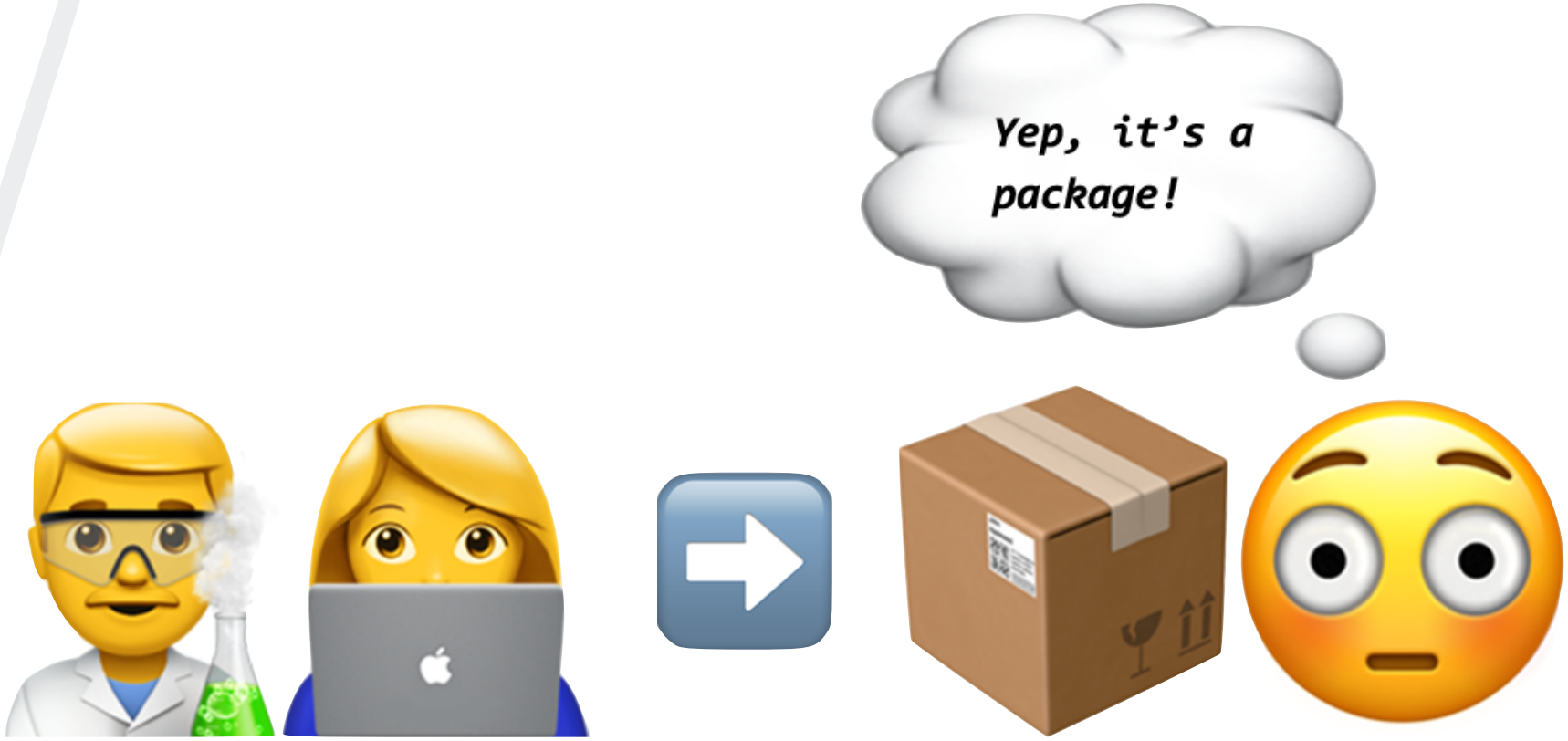
"Abstraction and Comprehension." Greg Wilson. <http://third-bit.com/2018/11/03/abstraction-comprehension.html> 2018-11-03.

Abstraction & comprehension



"Abstraction and Comprehension." Greg Wilson. <http://third-bit.com/2018/11/03/abstraction-comprehension.html> 2018-11-03.

require(n00bs)



"How rOpenSci uses Code Review to Promote reproducible Science." Noam Ross, Scott Chamberlain, Karthik Ram, Maëlle Salmon. <https://ropensci.org/blog/2017/09/01/nf-softwarereview/> 2017-09-01.

require(n00bs)

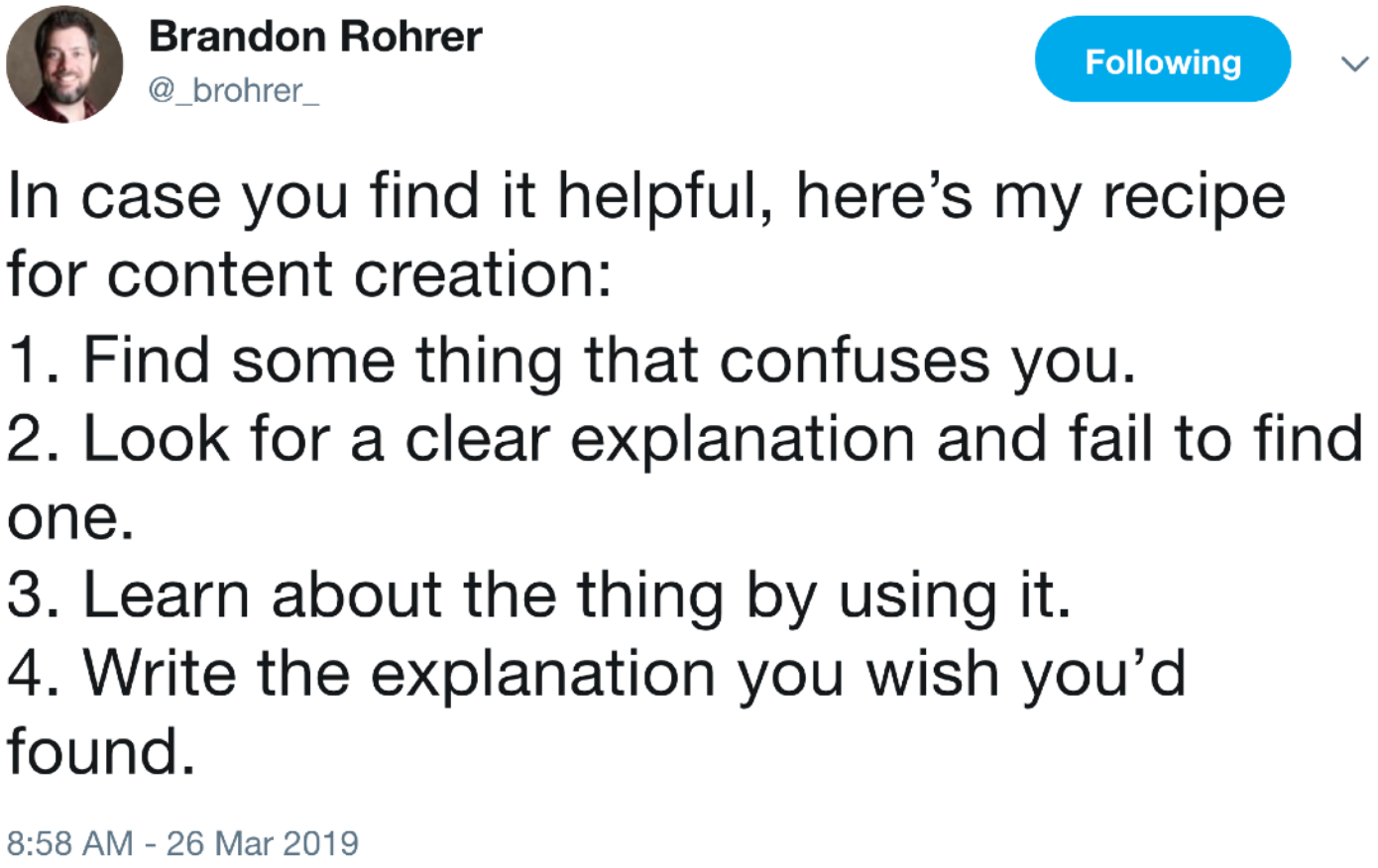
When assigning reviewers to a submission, we aim to pair experienced reviewers with new ones, or reviewers with expertise on a package's programming methods with those experienced in its field of application.

How rOpenSci uses Code Review to Promote Reproducible Science

ropensci.org highlighted with Highly

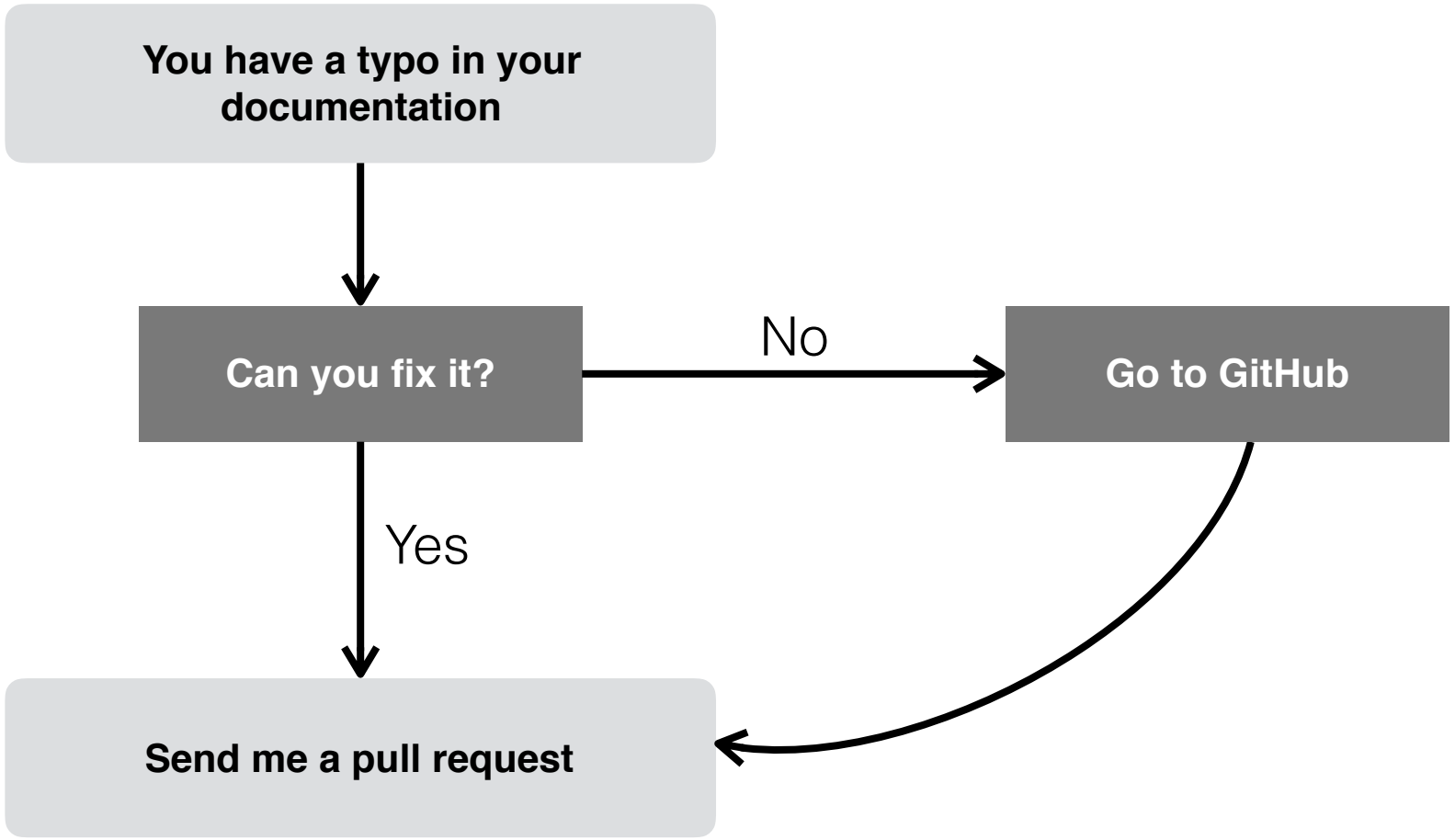
"How rOpenSci uses Code Review to Promote reproducible Science." Noam Ross, Scott Chamberlain, Karthik Ram, Maëlle Salmon. <https://ropensci.org/blog/2017/09/01/nf-softwarereview/> 2017-09-01.

Content creation



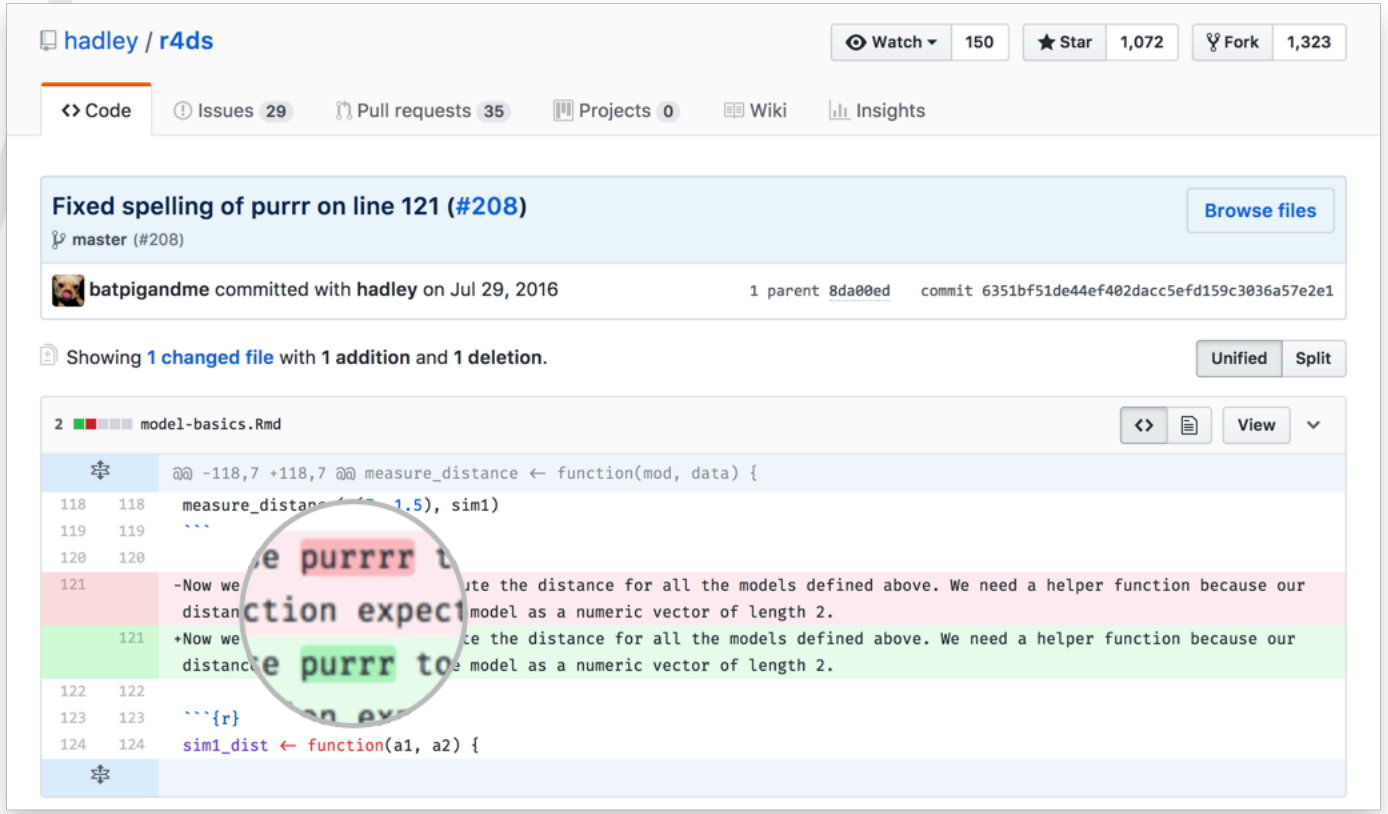
Brandon Rohrer <https://twitter.com/_brohrer_/status/1110526400933609472> 26 March 2019.

typos



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

My first "contribution"



Ways to contribute

PULL REQUESTS

Contributing code/making fixes.

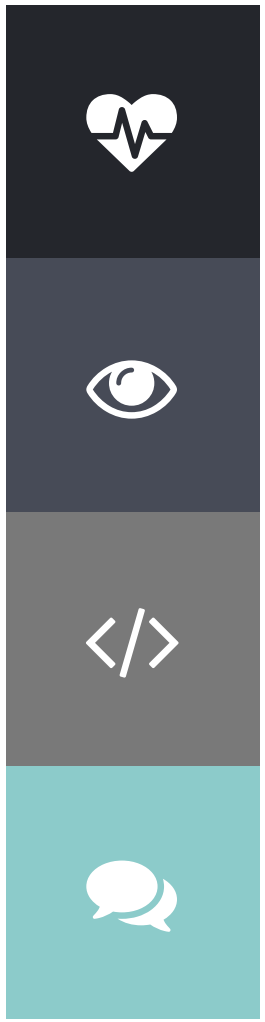
ISSUES

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).

Hints for happy code contributions in the tidyverse



GET THE PULSE OF A PROJECT

WATCH THE REPO

READ THE CODE

DISCUSS YOUR IDEAS

CONTRIBUTING.md

tidyverse & package conventions

KEY PACKAGES FOR CONTRIBUTING

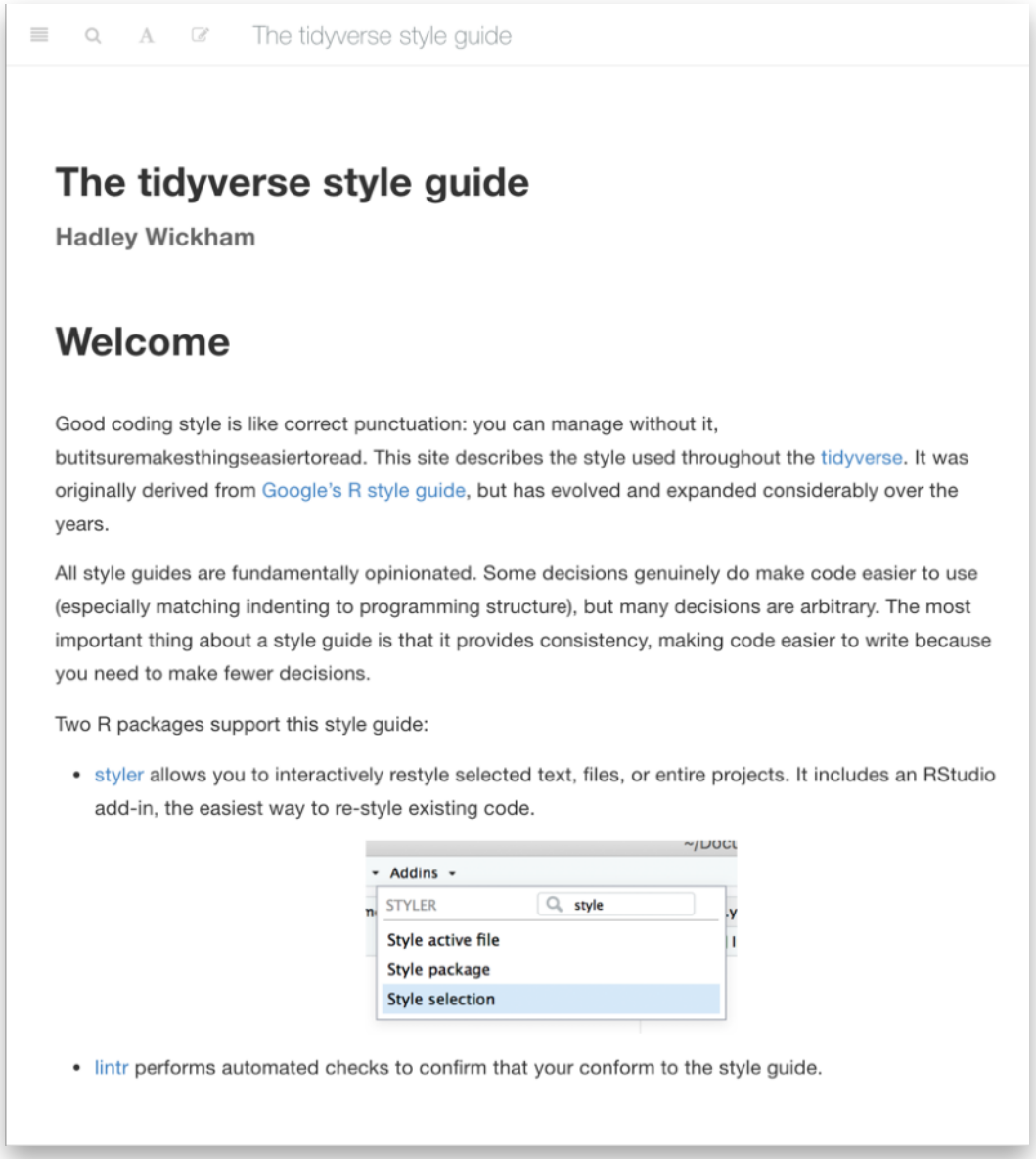
ROXYGEN2

DEVTOOLS

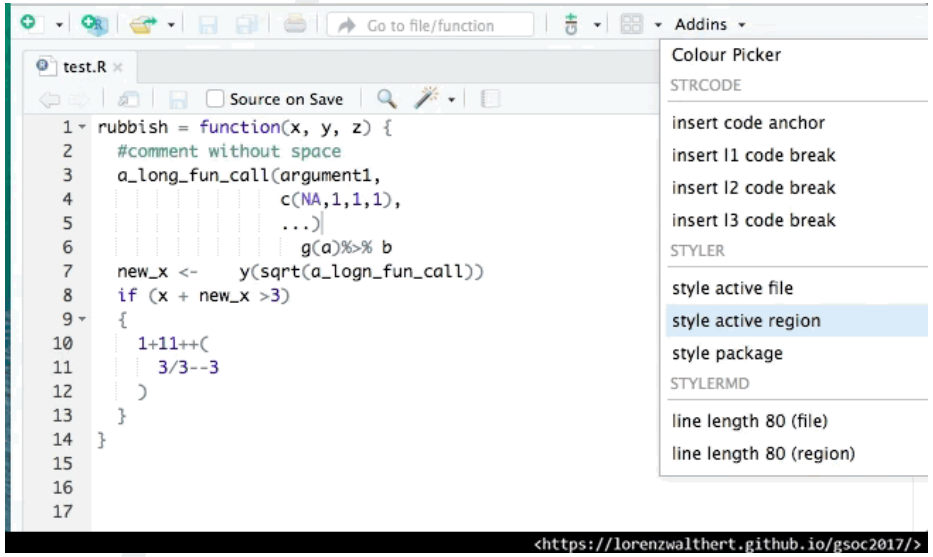
TESTTHAT

roxygen2: In-Line Documentation for R. R package v 6.0.1. by Hadley Wickham, Peter Danenberg and Manuel Eugster (2017). https://CRAN.R-project.org/package=roxygen2

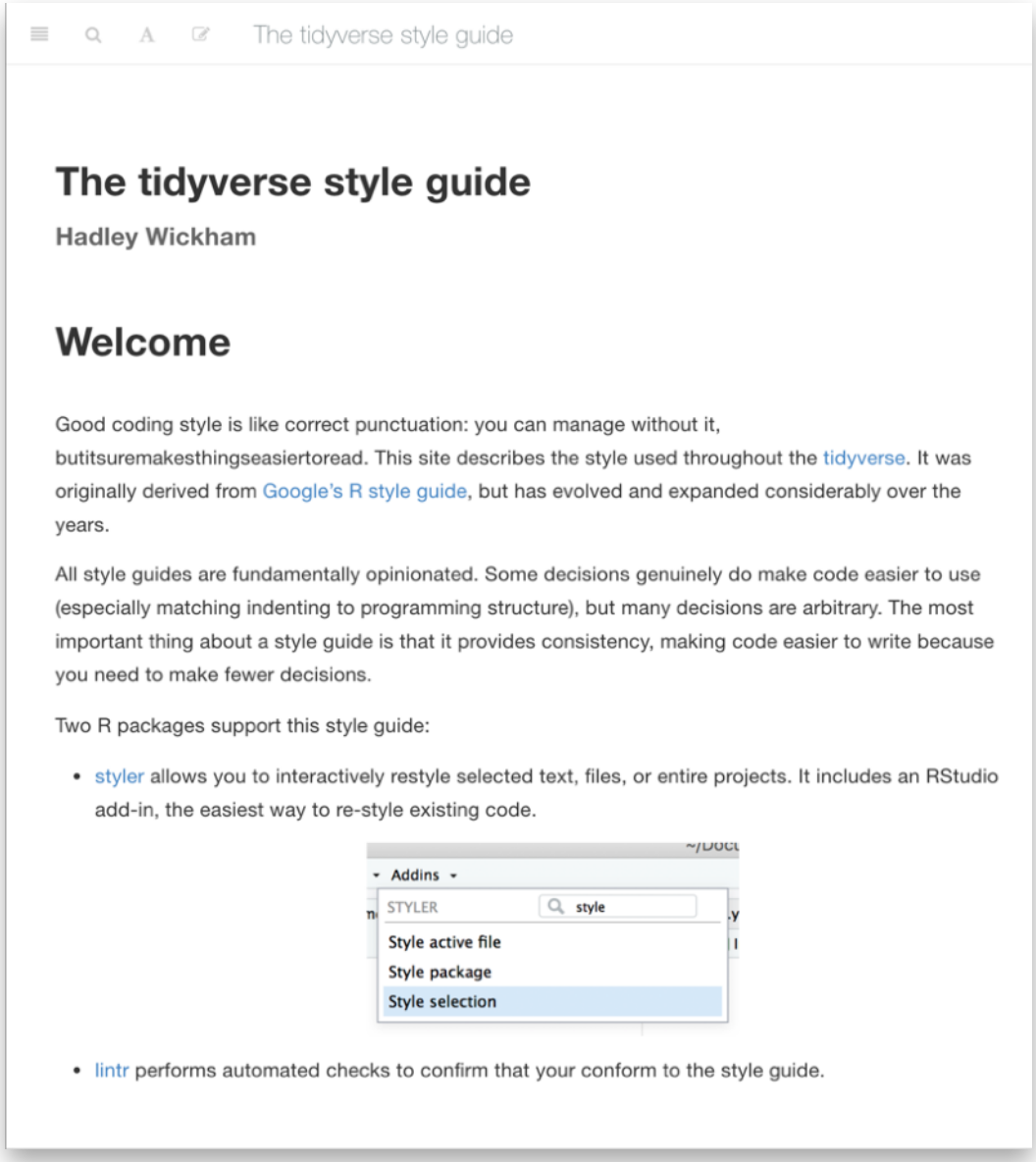
stylish...


The tidyverse style guide
The tidyverse style guide
Hadley Wickham
Welcome
Good coding style is like correct punctuation: you can manage without it, butitsuremakesthingseasiertoread. This site describes the style used throughout the tidyverse. It was originally derived from Google's R style guide, but has evolved and expanded considerably over the years.
All style guides are fundamentally opinionated. Some decisions genuinely do make code easier to use (especially matching indenting to programming structure), but many decisions are arbitrary. The most important thing about a style guide is that it provides consistency, making code easier to write because you need to make fewer decisions.
Two R packages support this style guide:
styler allows you to interactively restyle selected text, files, or entire projects. It includes an RStudio add-in, the easiest way to re-style existing code.
Addins
STYLER
style
Style active file
Style package
Style selection
lintr performs automated checks to confirm that your conform to the style guide.

stylish...



styler by Kirill Müller & Lorenz Walthert http://styler.r-lib.org



need help getting started?

Save the date: tidyverse developer day

Photo by rawpixel

Save the date 📆

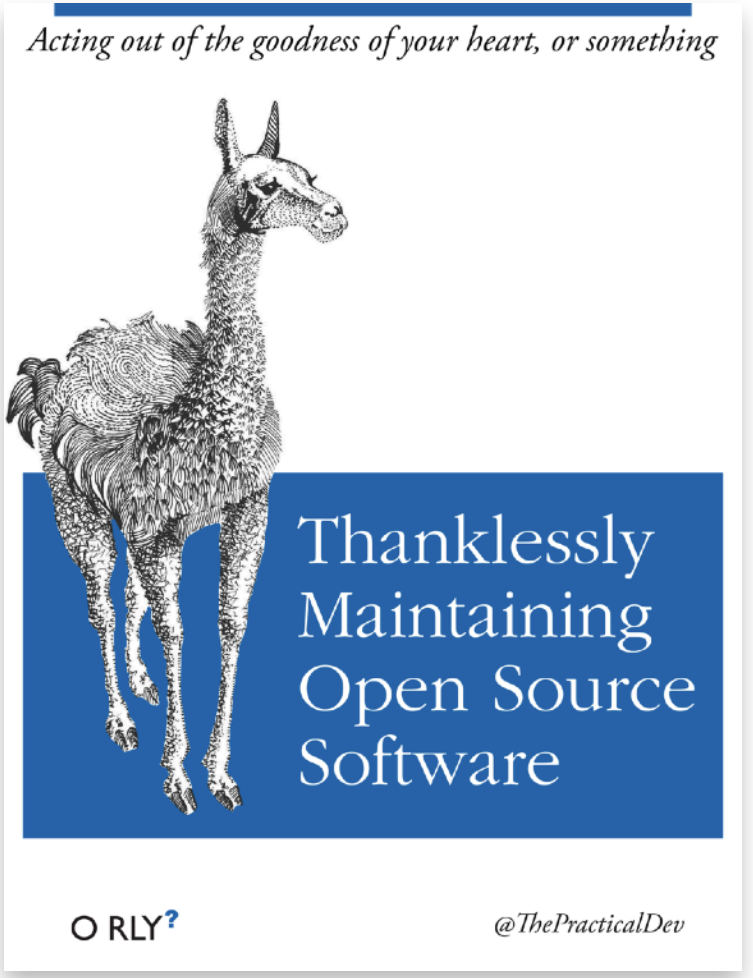
On the Saturday following rstudio::conf, we'll be holding our first ever **tidyverse developer day**, and you're invited! So, if you're interested, plan on being in Austin on the 19th of January. The venue is not settled yet, but the intent is to make it convenient for people who've chosen lodging based 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
Thanklessly Maintaining Open Source Software
O RLY?
@ThePracticalDev

tidyverse team

Thank You

Works cited

- Traweek, Sharon. 1988. Beamtimes and Lifetimes: The World of High Energy Physics. Cambridge, MA: Harvard University Press.
- Thieme, N. (2018). R generation. Significance, 15(4), 14–19. http://doi.org/10.1111/j.1740-9713.2018.01169.x
- Pintscher, Lydia, ed. 2012. *Open Advice: FOSS: What We Wish We Had Known When We Started*. http://open-advice.org/
- Hawthorn, L. (2012). You'll Eventually Know Everything They've Forgotten. In L. Pintscher (Ed.), *Open Advice: FOSS: What We Wish We Had Known When We Started* (pp. 29–32).
- Macieira, T. (2012). The Art of Problem Solving. In L. Pintscher (Ed.), *Open Advice: FOSS: What We Wish We Had Known When We Started* (pp. 55–61).
- 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
- 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
- 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