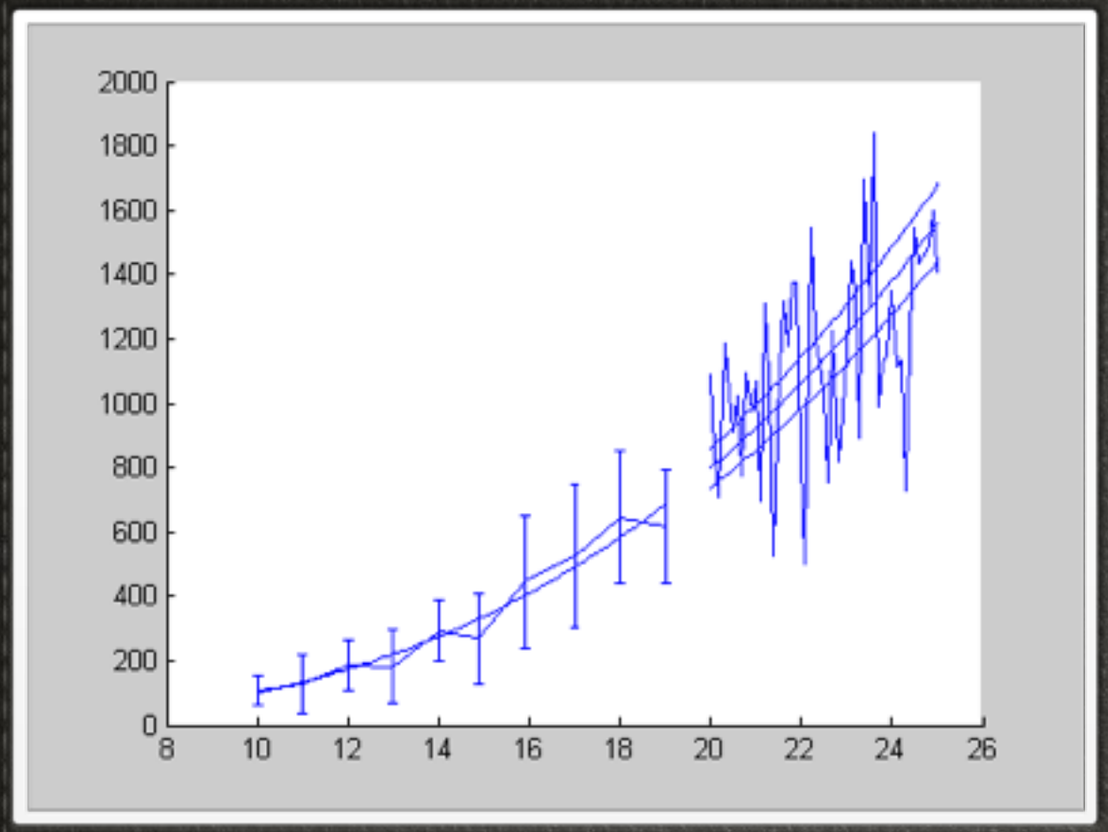
MAKING PLOTS PRETTY

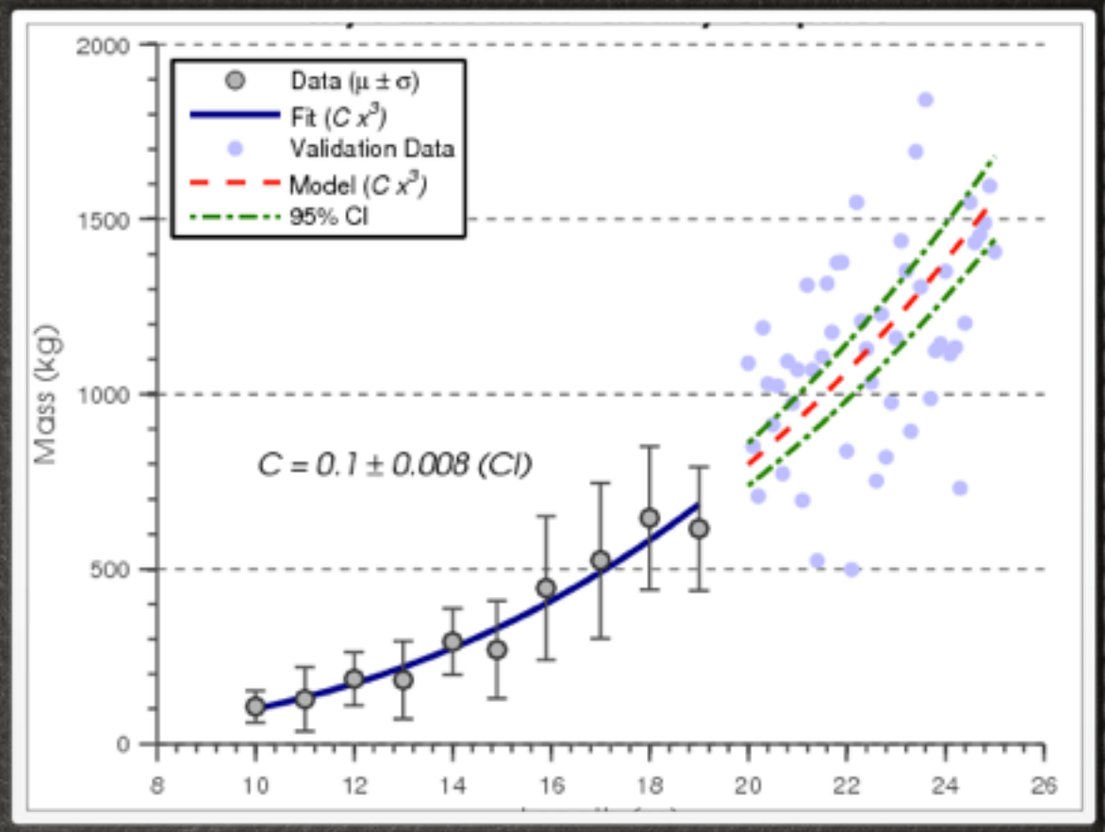
or: What you don't know probably won't hurt you, but could be turned into something that might help you if you knew it (maybe)

Max Forbes — HCR Lab Meeting 4/16/2014

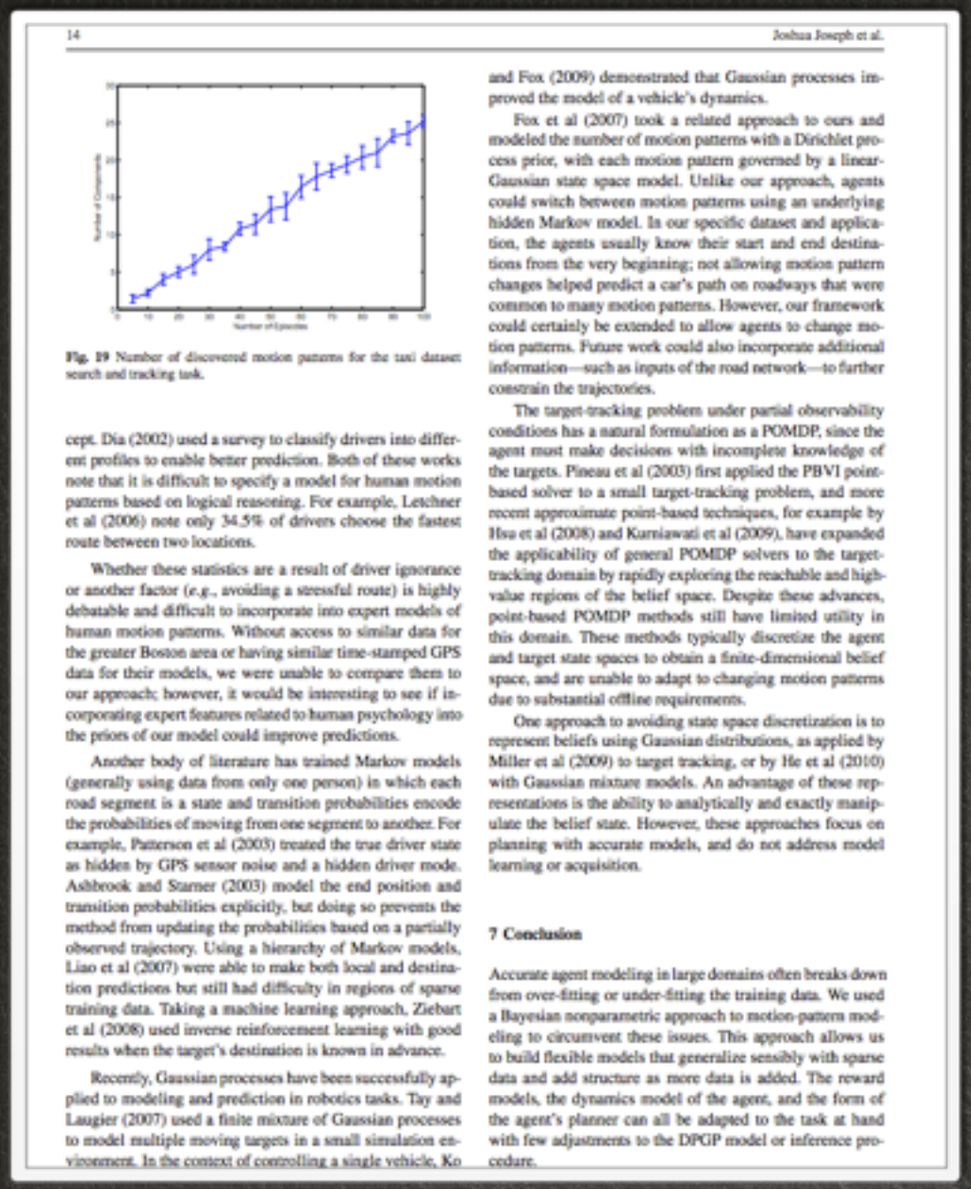
1. MOTIVATION

The obligatory big picture.





1. TELL A MORE COMPELLING STORY



14 Joshua Joseph et al.

Fig. 19 Number of discovered motion patterns for the taxi dataset search and tracking task.

cept. Dia (2002) used a survey to classify drivers into different profiles to enable better prediction. Both of these works note that it is difficult to specify a model for human motion patterns based on logical reasoning. For example, Letchner et al (2006) note only 34.5% of drivers choose the fastest route between two locations.

Whether these statistics are a result of driver ignorance or another factor (*e.g.*, avoiding a stressful route) is highly debatable and difficult to incorporate into expert models of human motion patterns. Without access to similar data for the greater Boston area or having similar time-stamped GPS data for their models, we were unable to compare them to our approach; however, it would be interesting to see if incorporating expert features related to human psychology into the priors of our model could improve predictions.

Another body of literature has trained Markov models (generally using data from only one person) in which each road segment is a state and transition probabilities encode the probabilities of moving from one segment to another. For example, Patterson et al (2003) treated the true driver state as hidden by GPS sensor noise and a hidden driver mode. Ashbrook and Starner (2003) model the end position and transition probabilities explicitly, but doing so prevents the method from updating the probabilities based on a partially observed trajectory. Using a hierarchy of Markov models, Liao et al (2007) were able to make both local and destination predictions but still had difficulty in regions of sparse training data. Taking a machine learning approach, Ziebart et al (2008) used inverse reinforcement learning with good results when the target's destination is known in advance.

Recently, Gaussian processes have been successfully applied to modeling and prediction in robotics tasks. Tay and Laugier (2007) used a finite mixture of Gaussian processes to model multiple moving targets in a small simulation environment. In the context of controlling a single vehicle, Ko and Fox (2009) demonstrated that Gaussian processes improved the model of a vehicle's dynamics.

Fox et al (2007) took a related approach to ours and modeled the number of motion patterns with a Dirichlet process prior, with each motion pattern governed by a linear-Gaussian state space model. Unlike our approach, agents could switch between motion patterns using an underlying hidden Markov model. In our specific dataset and application, the agents usually know their start and end destinations from the very beginning; not allowing motion pattern changes helped predict a car's path on roadways that were common to many motion patterns. However, our framework could certainly be extended to allow agents to change motion patterns. Future work could also incorporate additional information—such as inputs of the road network—to further constrain the trajectories.

The target-tracking problem under partial observability conditions has a natural formulation as a POMDP, since the agent must make decisions with incomplete knowledge of the targets. Pineau et al (2003) first applied the PBVI point-based solver to a small target-tracking problem, and more recent approximate point-based techniques, for example by Hsu et al (2008) and Kurniawati et al (2009), have expanded the applicability of general POMDP solvers to the target-tracking domain by rapidly exploring the reachable and high-value regions of the belief space. Despite these advances, point-based POMDP methods still have limited utility in this domain. These methods typically discretize the agent and target state spaces to obtain a finite-dimensional belief space, and are unable to adapt to changing motion patterns due to substantial offline requirements.

One approach to avoiding state space discretization is to represent beliefs using Gaussian distributions, as applied by Miller et al (2009) to target tracking, or by He et al (2010) with Gaussian mixture models. An advantage of these representations is the ability to analytically and exactly manipulate the belief state. However, these approaches focus on planning with accurate models, and do not address model learning or acquisition.

7 Conclusion

Accurate agent modeling in large domains often breaks down from over-fitting or under-fitting the training data. We used a Bayesian nonparametric approach to motion-pattern modeling to circumvent these issues. This approach allows us to build flexible models that generalize sensibly with sparse data and add structure as more data is added. The reward models, the dynamics model of the agent, and the form of the agent's planner can all be adapted to the task at hand with few adjustments to the DPGP model or inference procedure.

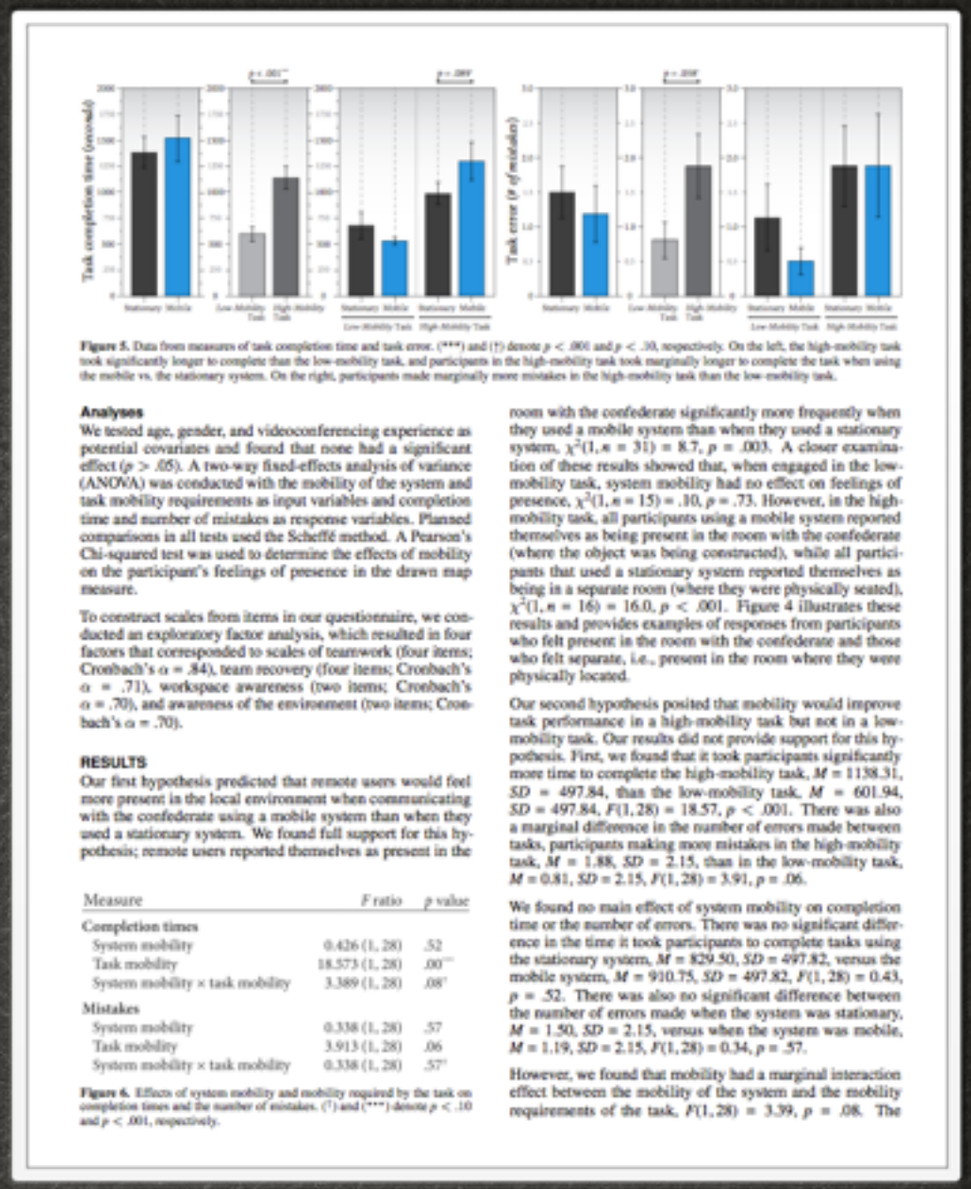


Figure 5. Data from measures of task completion time and task error. (***) and (†) denote $p < .001$ and $p < .10$, respectively. On the left, the high-mobility task took significantly longer to complete than the low-mobility task, and participants in the high-mobility task took marginally longer to complete the task when using the mobile vs. the stationary system. On the right, participants made marginally more mistakes in the high-mobility task than the low-mobility task.

Analyses

We tested age, gender, and videoconferencing experience as potential covariates and found that none had a significant effect ($p > .05$). A two-way fixed-effects analysis of variance (ANOVA) was conducted with the mobility of the system and task mobility requirements as input variables and completion time and number of mistakes as response variables. Planned comparisons in all tests used the Scheffé method. A Pearson's Chi-squared test was used to determine the effects of mobility on the participant's feelings of presence in the drawn map measure.

To construct scales from items in our questionnaire, we conducted an exploratory factor analysis, which resulted in four factors that corresponded to scales of teamwork (four items; Cronbach's $\alpha = .84$), team recovery (four items; Cronbach's $\alpha = .71$), workspace awareness (two items; Cronbach's $\alpha = .70$), and awareness of the environment (two items; Cronbach's $\alpha = .70$).

RESULTS

Our first hypothesis predicted that remote users would feel more present in the local environment when communicating with the confederate using a mobile system than when they used a stationary system. We found full support for this hypothesis; remote users reported themselves as present in the room with the confederate significantly more frequently when they used a mobile system than when they used a stationary system, $\chi^2(1, n = 31) = 8.7$, $p = .003$. A closer examination of these results showed that, when engaged in the low-mobility task, system mobility had no effect on feelings of presence, $\chi^2(1, n = 15) = .10$, $p = .73$. However, in the high-mobility task, all participants using a mobile system reported themselves as being present in the room with the confederate (where the object was being constructed), while all participants that used a stationary system reported themselves as being in a separate room (where they were physically seated), $\chi^2(1, n = 16) = 16.0$, $p < .001$. Figure 4 illustrates these results and provides examples of responses from participants who felt present in the room with the confederate and those who felt separate, i.e., present in the room where they were physically located.

Our second hypothesis posited that mobility would improve task performance in a high-mobility task but not in a low-mobility task. Our results did not provide support for this hypothesis. First, we found that it took participants significantly more time to complete the high-mobility task, $M = 1138.31$, $SD = 497.84$, than the low-mobility task, $M = 601.94$, $SD = 497.84$, $F(1, 28) = 18.57$, $p < .001$. There was also a marginal difference in the number of errors made between tasks, participants making more mistakes in the high-mobility task, $M = 1.88$, $SD = 2.15$, than in the low-mobility task, $M = 0.81$, $SD = 2.15$, $F(1, 28) = 3.91$, $p = .06$.

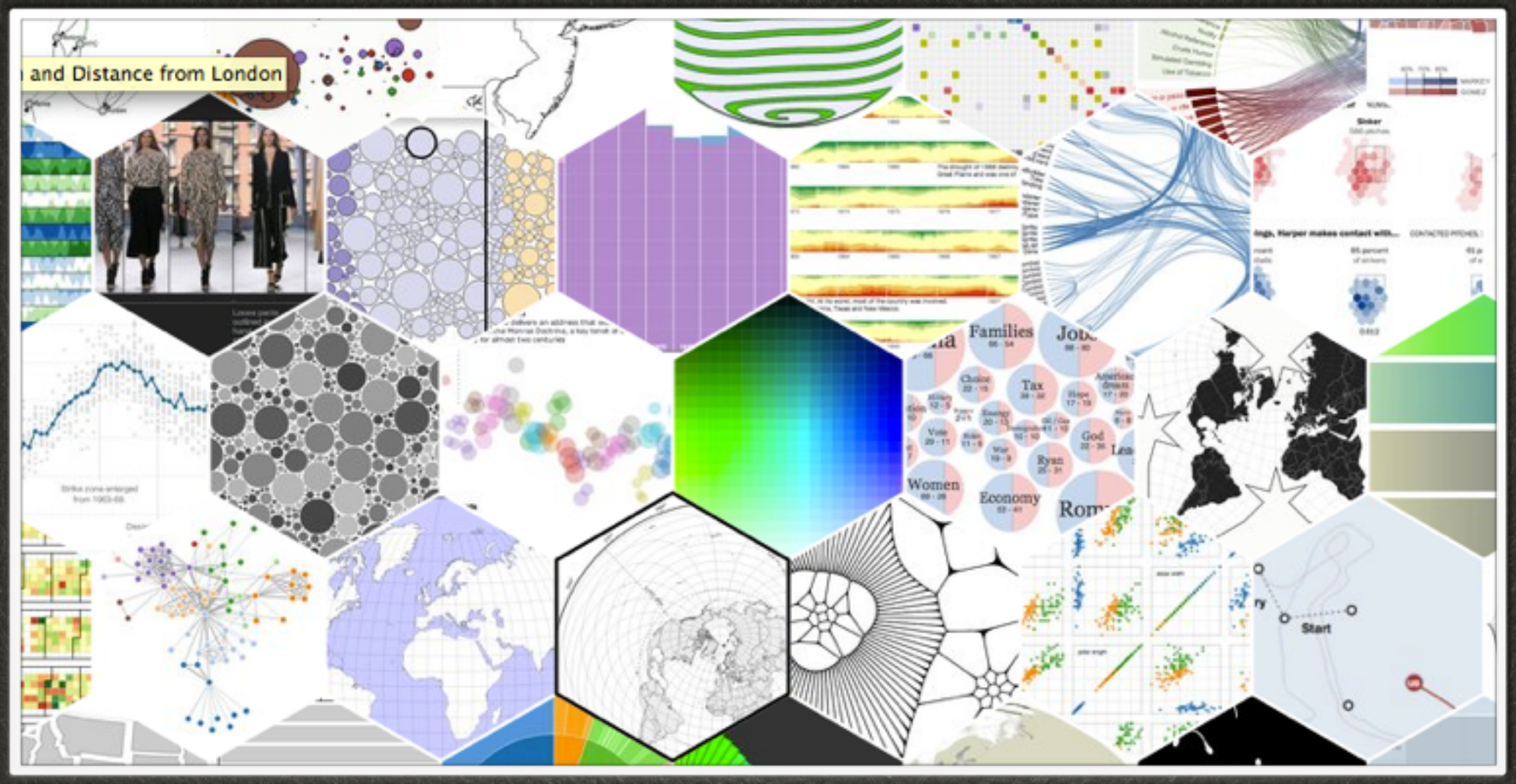
We found no main effect of system mobility on completion time or the number of errors. There was no significant difference in the time it took participants to complete tasks using the stationary system, $M = 829.50$, $SD = 497.82$, versus the mobile system, $M = 910.75$, $SD = 497.82$, $F(1, 28) = 0.43$, $p = .52$. There was also no significant difference between the number of errors made when the system was stationary, $M = 1.50$, $SD = 2.15$, versus when the system was mobile, $M = 1.19$, $SD = 2.15$, $F(1, 28) = 0.34$, $p = .57$.

However, we found that mobility had a marginal interaction effect between the mobility of the system and the mobility requirements of the task, $F(1, 28) = 3.39$, $p = .08$. The

Measure	*F* ratio	*p* value
Completion times		
System mobility	0.426 (1, 28)	.52
Task mobility	18.573 (1, 28)	.00***
System mobility × task mobility	3.389 (1, 28)	.08†
Mistakes		
System mobility	0.338 (1, 28)	.57
Task mobility	3.913 (1, 28)	.06
System mobility × task mobility	0.338 (1, 28)	.57†

Figure 6. Effects of system mobility and mobility required by the task on completion times and the number of mistakes. (†) and (***) denote $p < .10$ and $p < .001$, respectively.

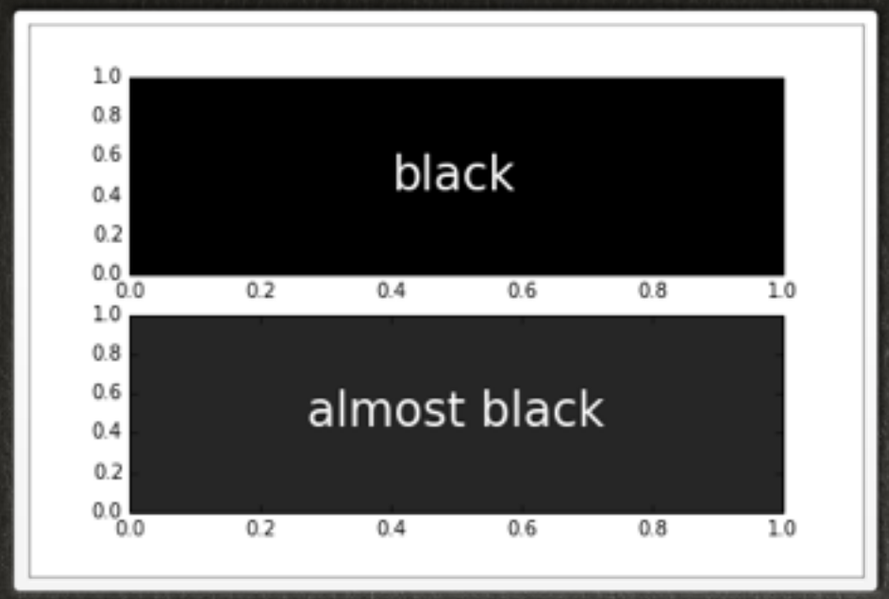
2. MAKE PAPERS BETTER-LOOKING

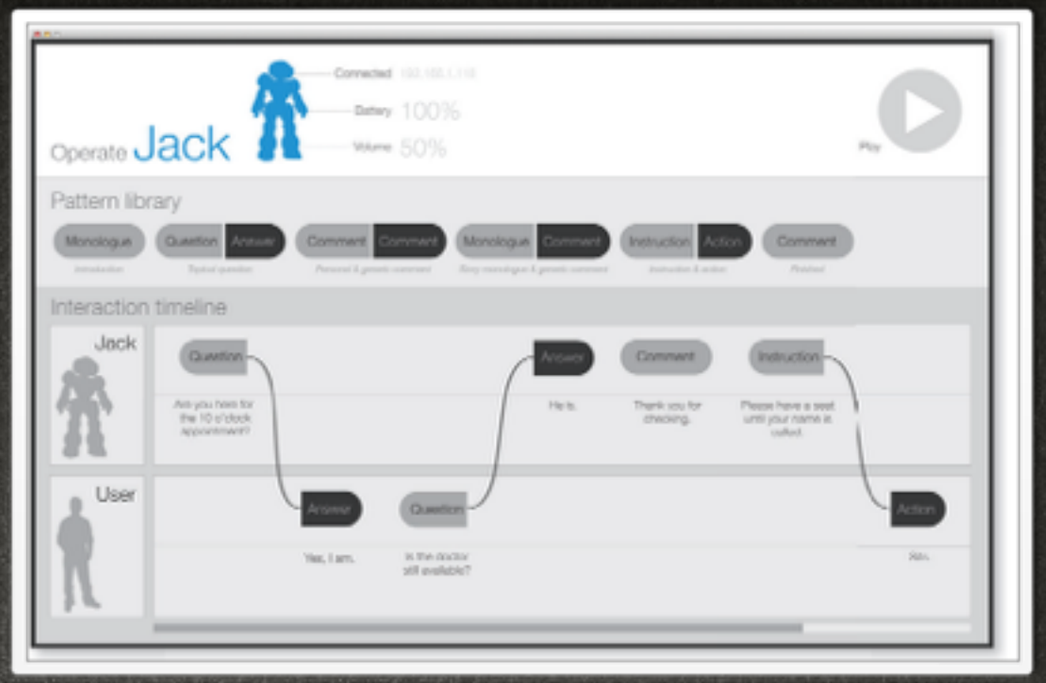


3. DATA VIZ IS COOL/ BEAUTIFUL/FUN/INSPIRING

2. TIPS AND TECHNIQUES

Tools for the short-of-attention-span.

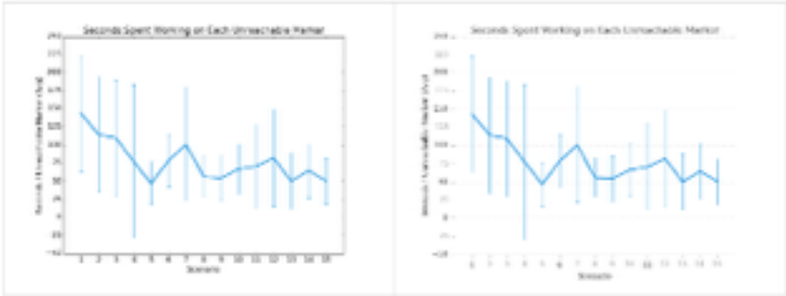




README.md

beautyplot

A single function call to make `matplotlib.pyplot` plots look better. Because neither prettyplotlib nor matplotlibrc worked for me.



Setup

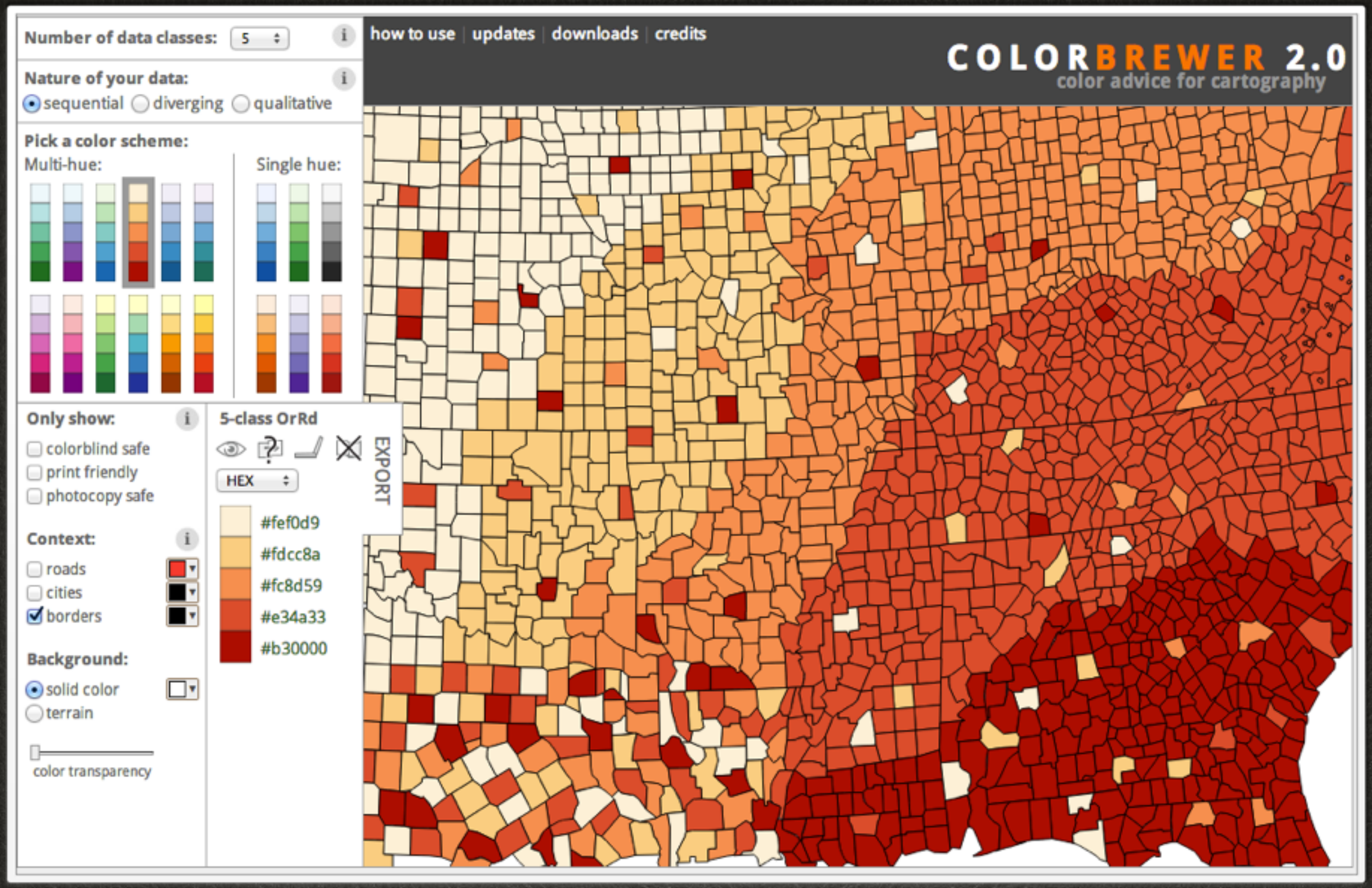
Add `beautyplot.py` to your `PYTHONPATH` so that Python can find it. I've provided a script that does this called `setup.sh`. You can run it (on Unix-like systems) by typing the following in a terminal:

```
source setup.sh
```

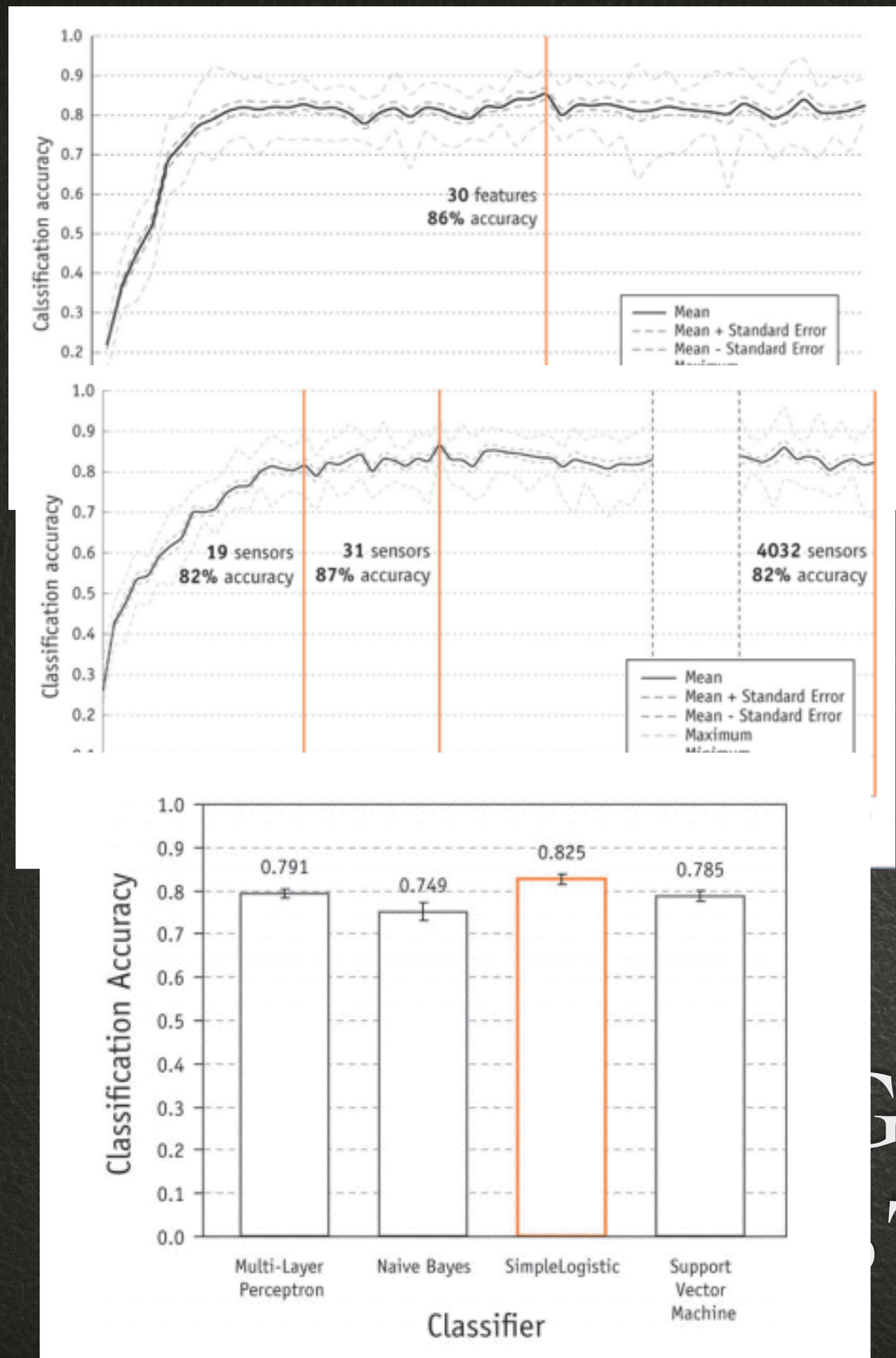
To use it regularly, I recommend you add what's in `setup.sh` it to your `.bashrc` or the like.

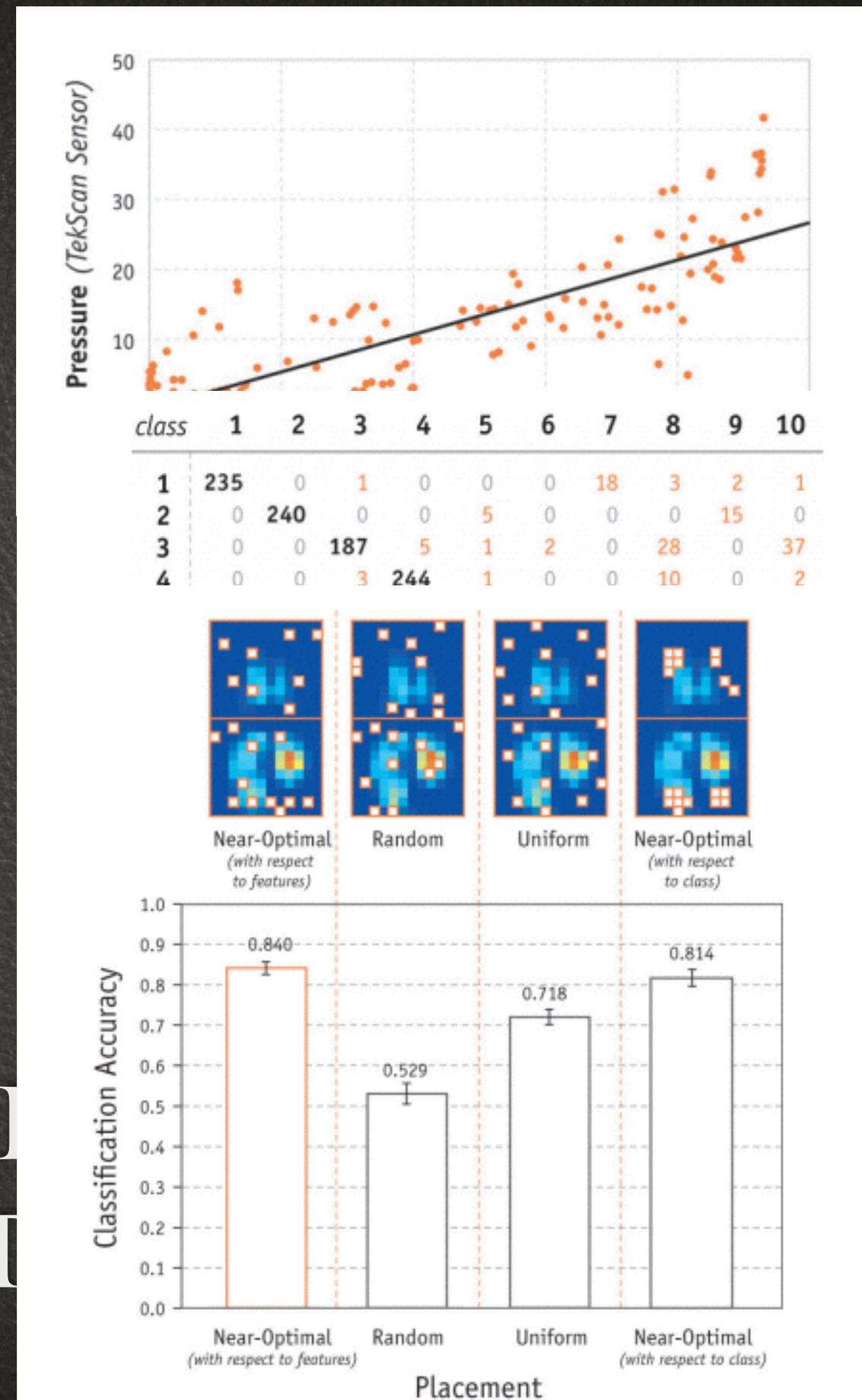
Usage

DON'T USE PURE BLACK



USE THE RIGHT COLORS





class	1	2	3	4	5	6	7	8	9	10
1	235	0	1	0	0	0	18	3	2	1
2	0	240	0	0	5	0	0	0	15	0
3	0	0	187	5	1	2	0	28	0	37
4	0	0	3	244	1	0	0	10	0	2

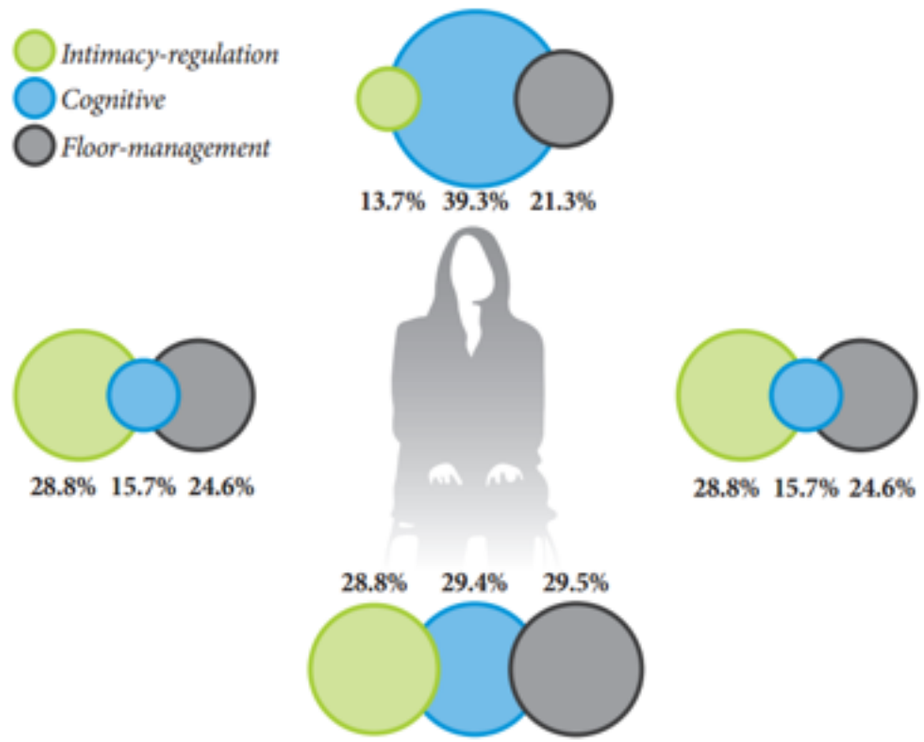
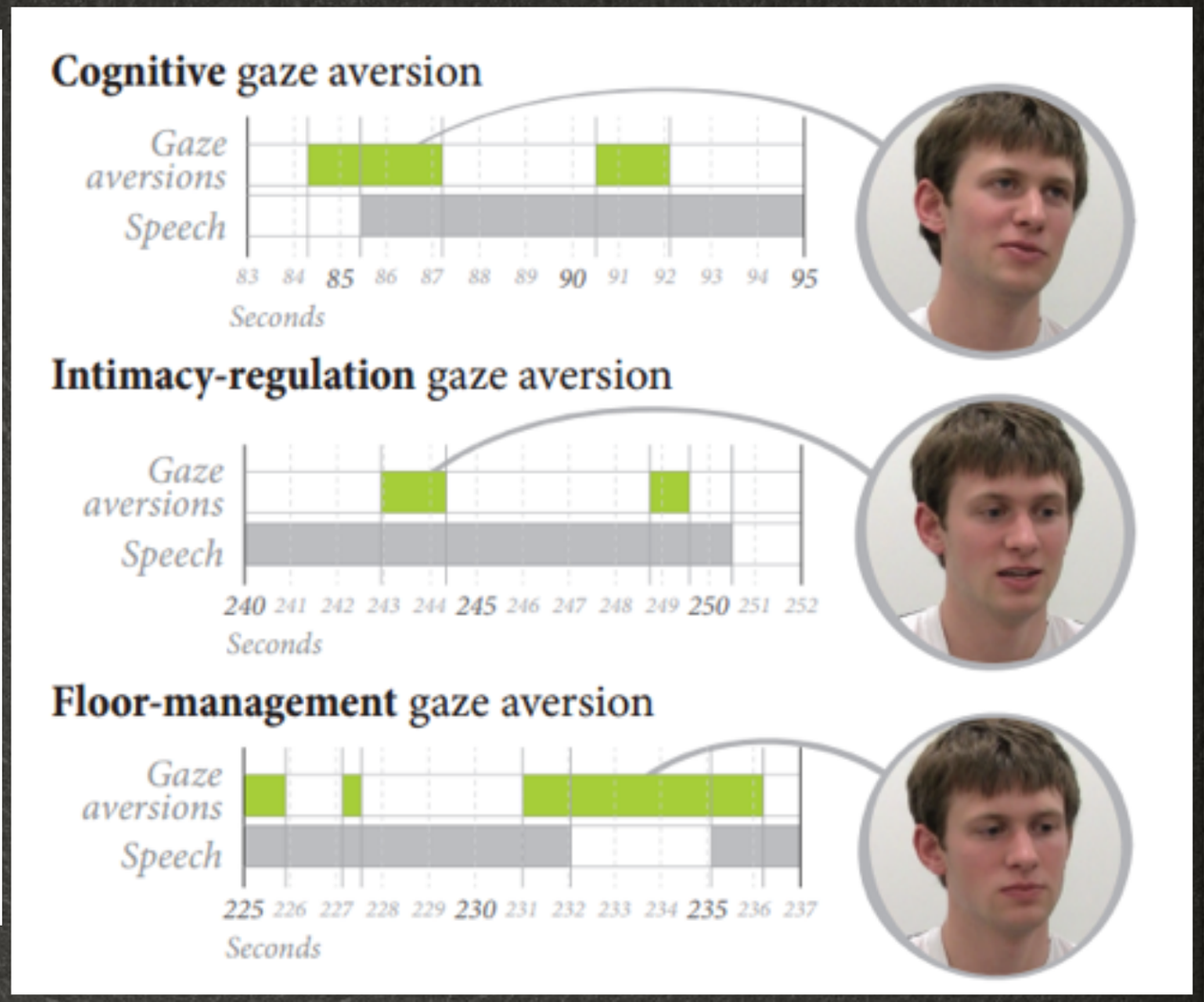
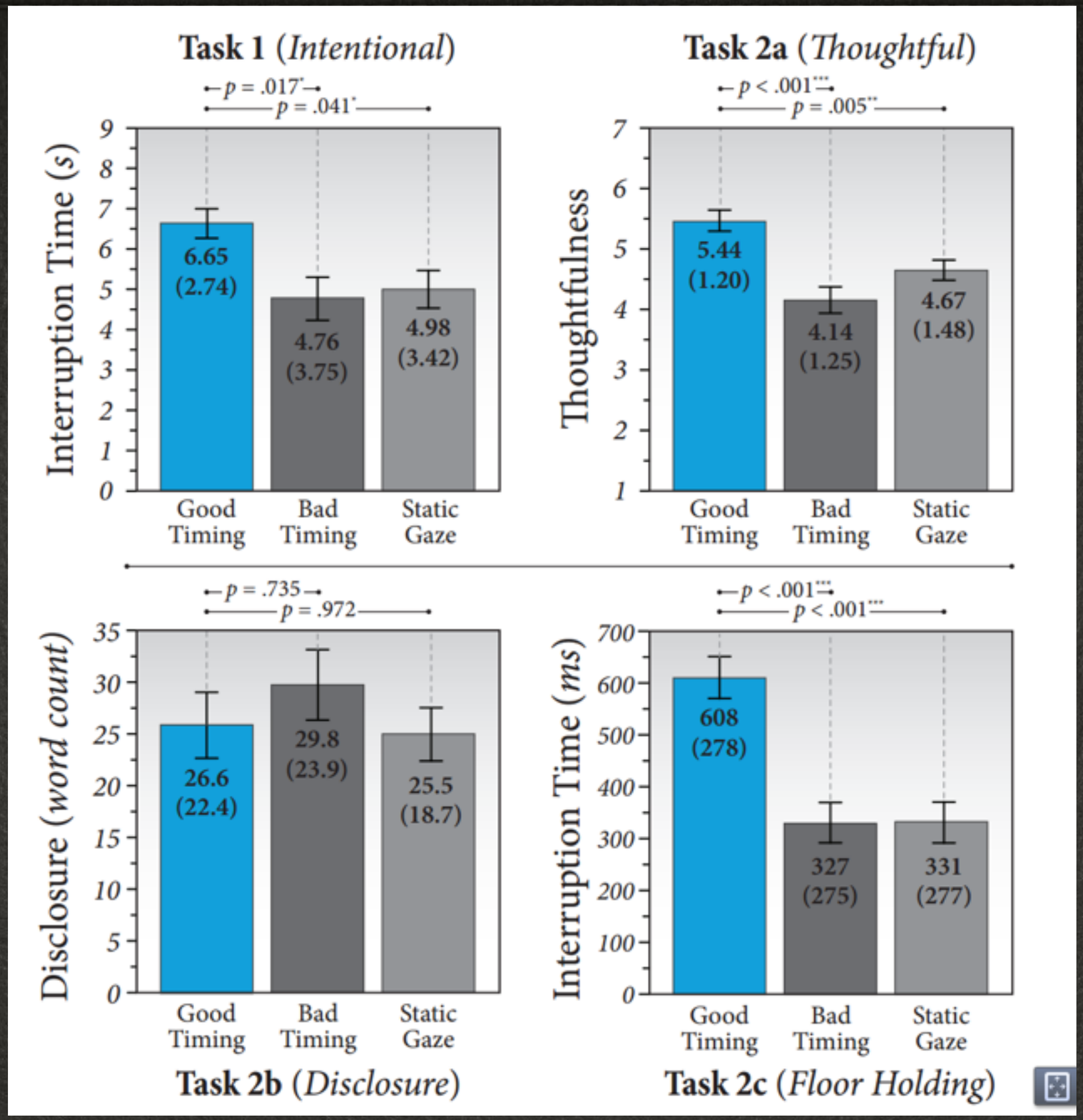


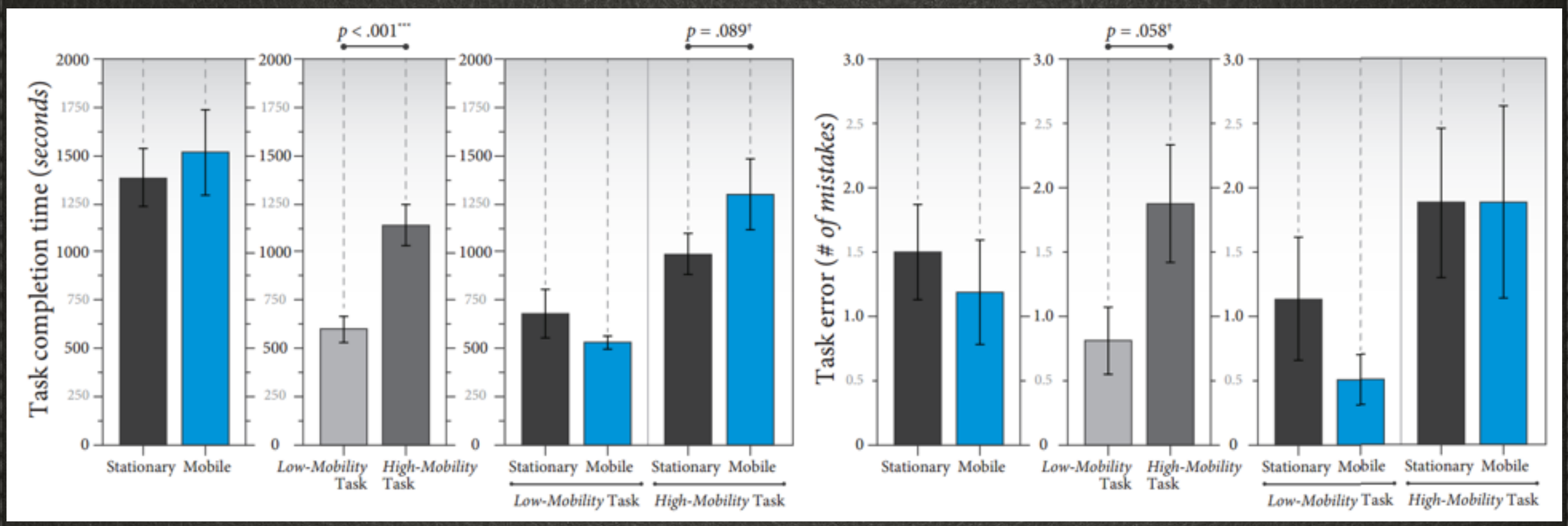
Figure 4: Percentages of gaze aversions directed up, down, and to the side, split by conversational function. Intimacy-regulating and floor-managing gaze aversions are more likely to be directed sideways, while cognitive gaze aversions are more likely to be directed upwards.



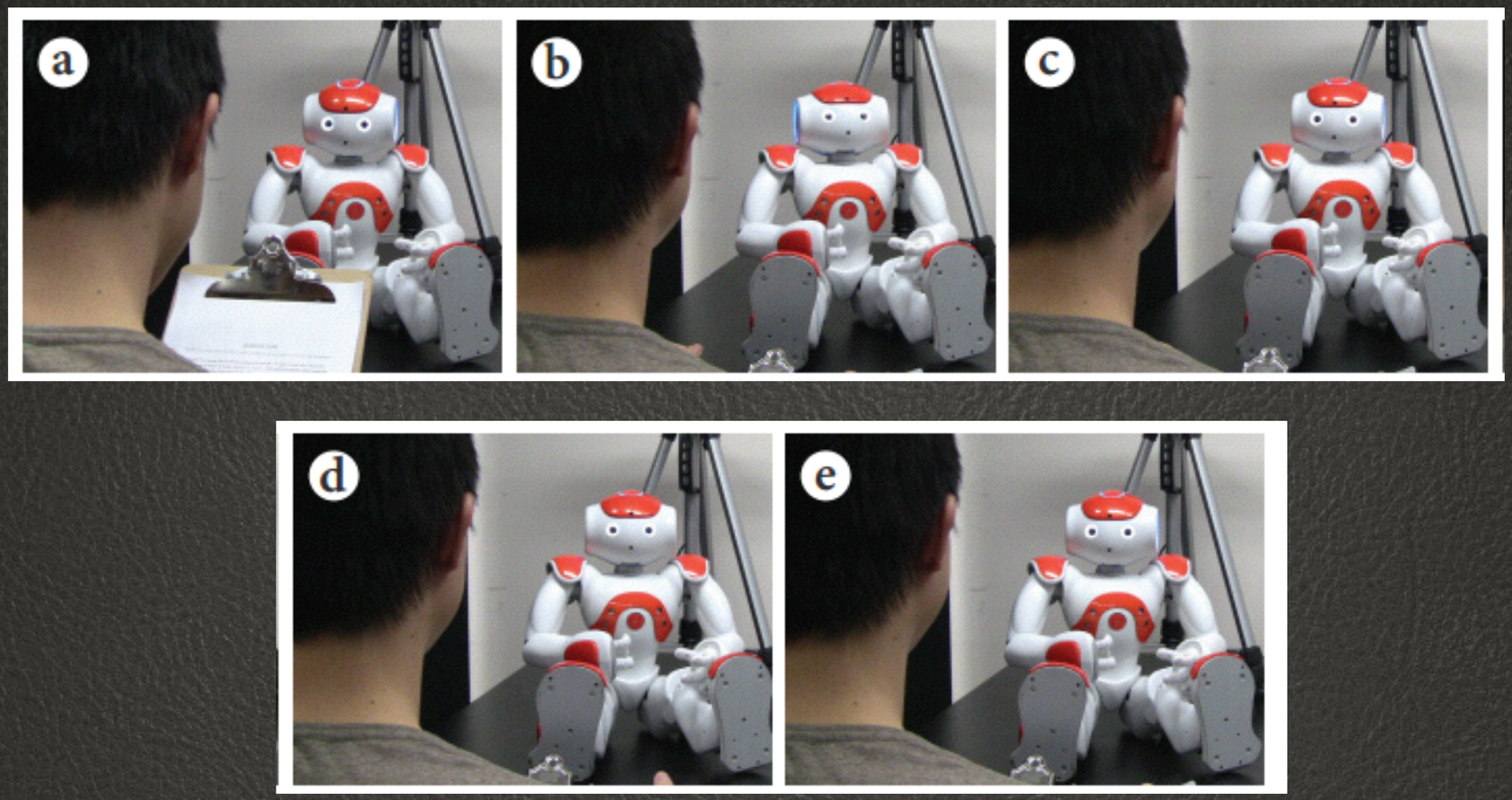
USE COLORS SPARINGLY



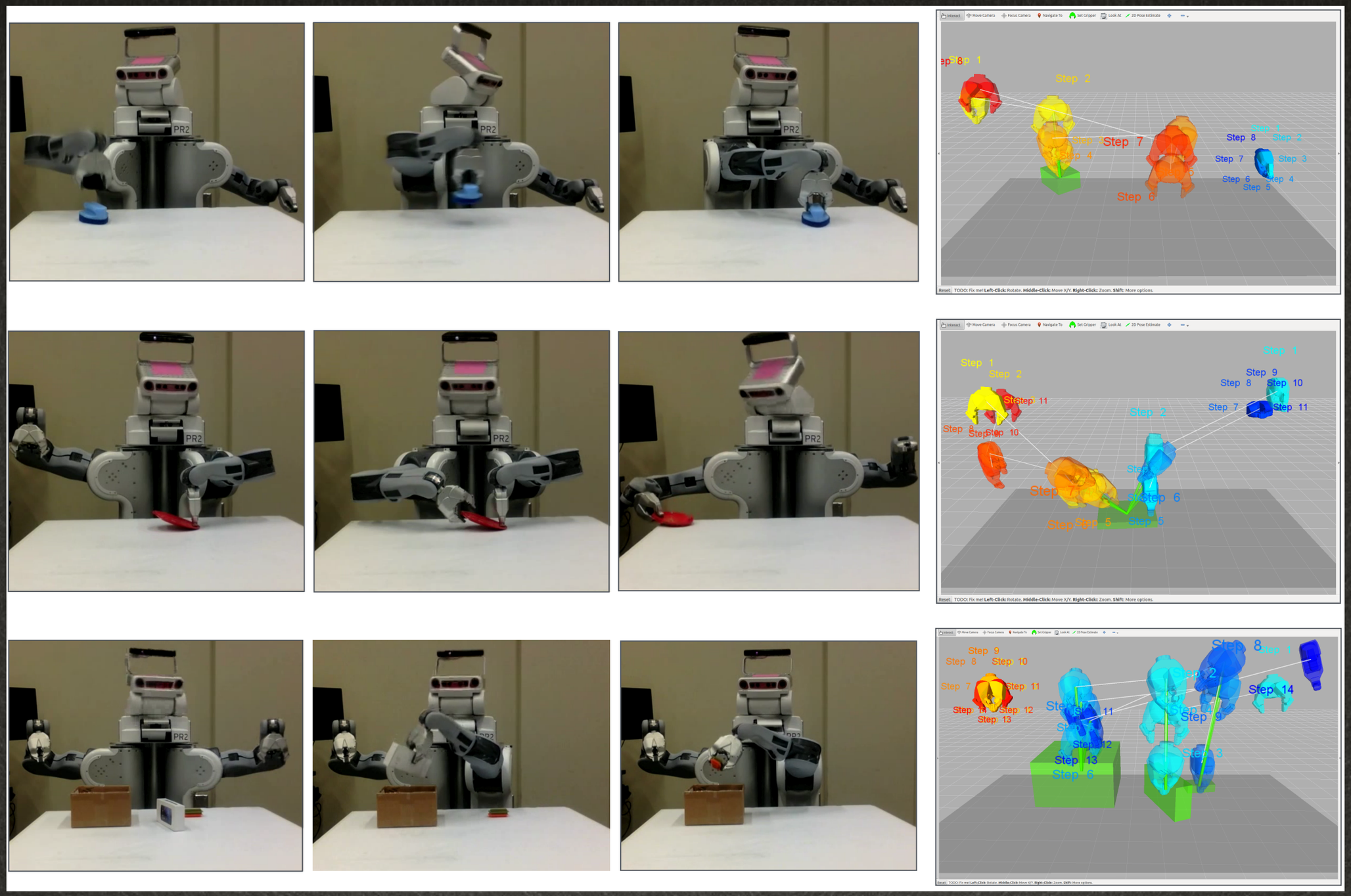
VALUE INFORMATION DENSITY



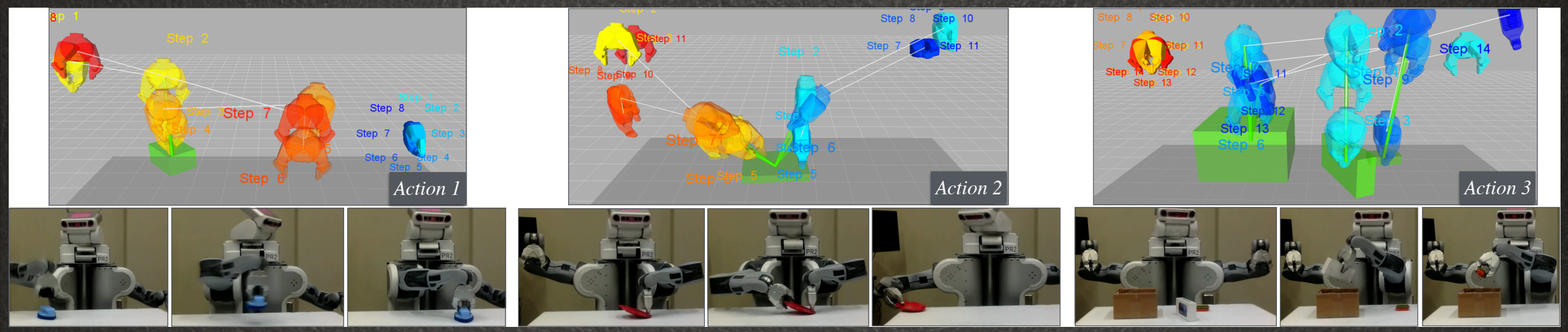
VALUE INFORMATION DENSITY



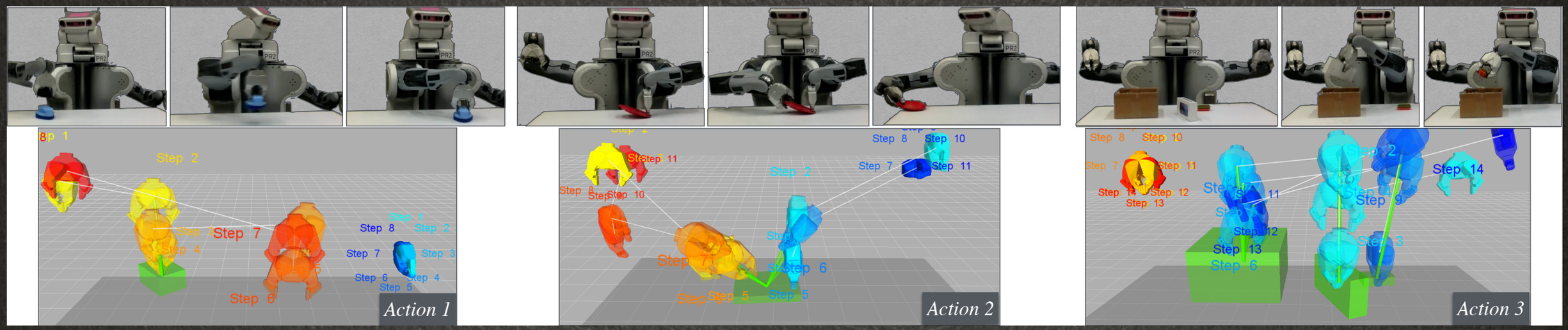
SIDE NOTE ON PICTURES: CROP WELL (PART 1)



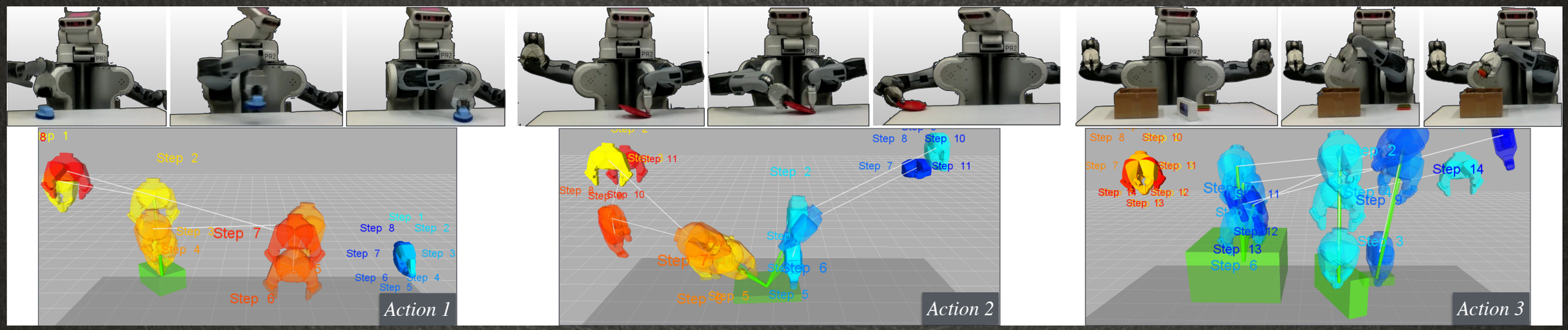
SIDE NOTE ON PICTURES: CROP WELL (PART 2)



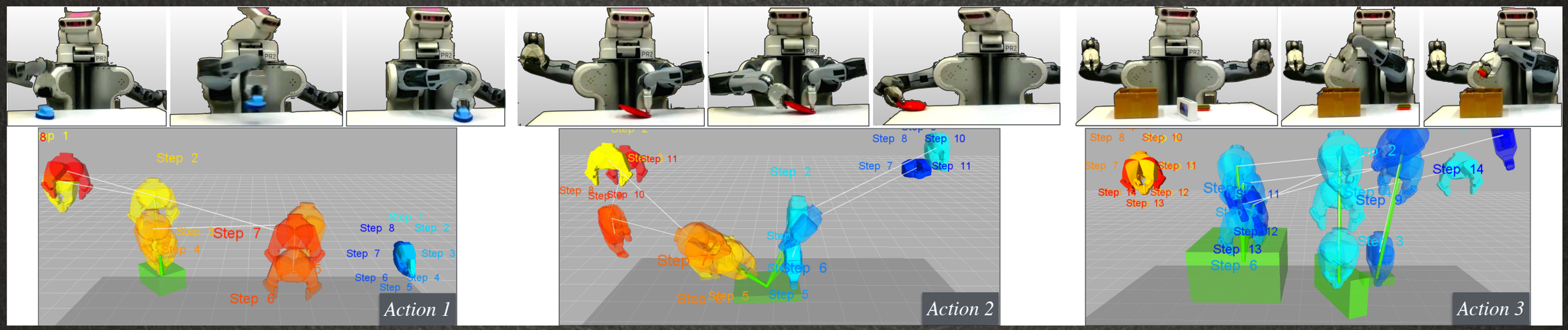
SIDE NOTE ON PICTURES: CROP WELL (PART 2)



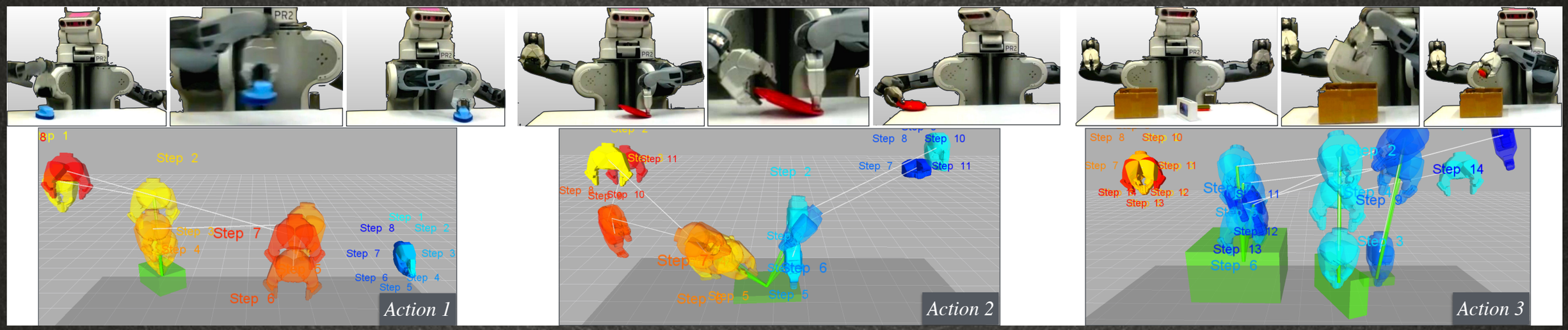
SIDE NOTE ON PICTURES: CROP WELL (PART 2)



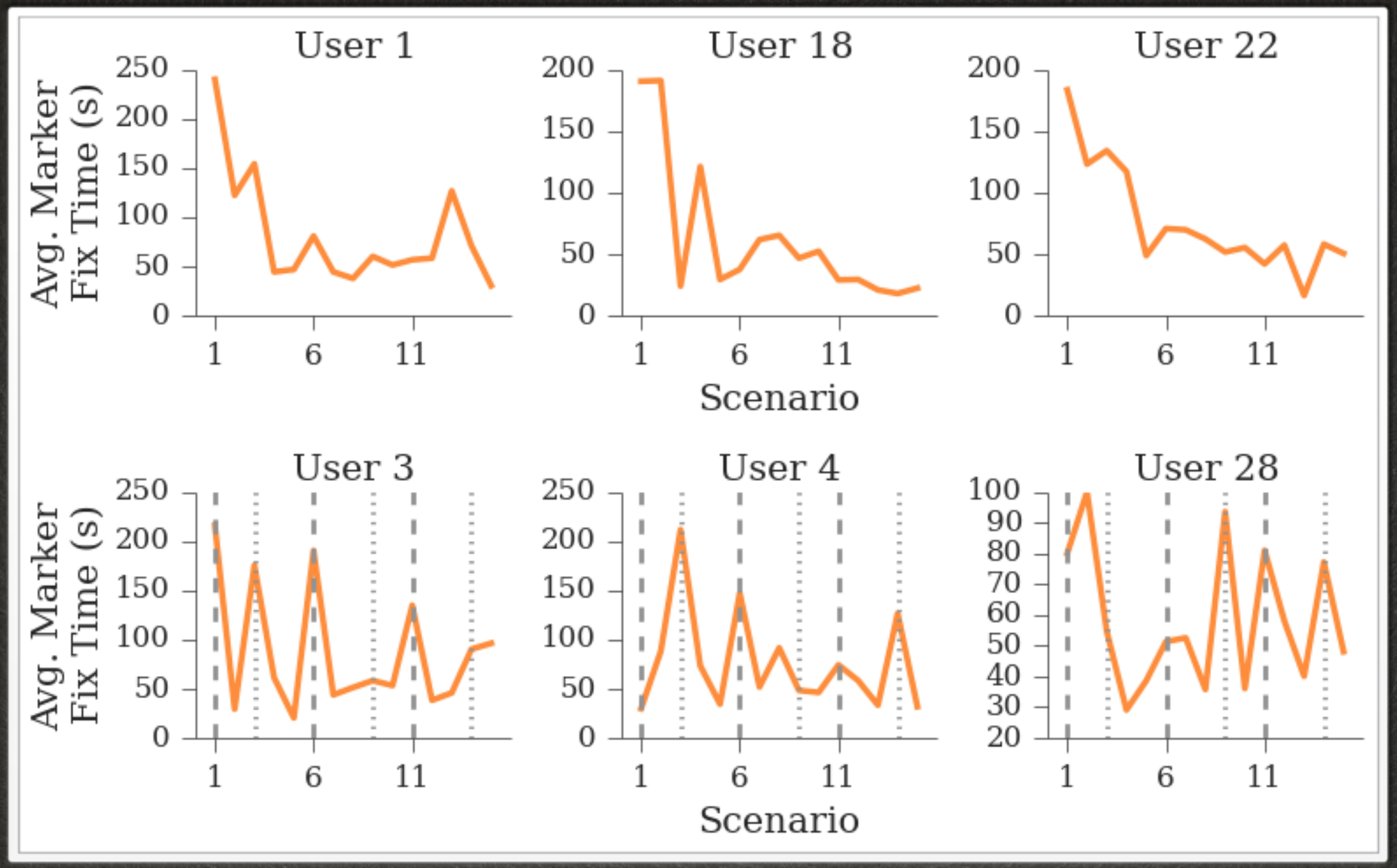
SIDE NOTE ON PICTURES: CROP WELL (PART 2)



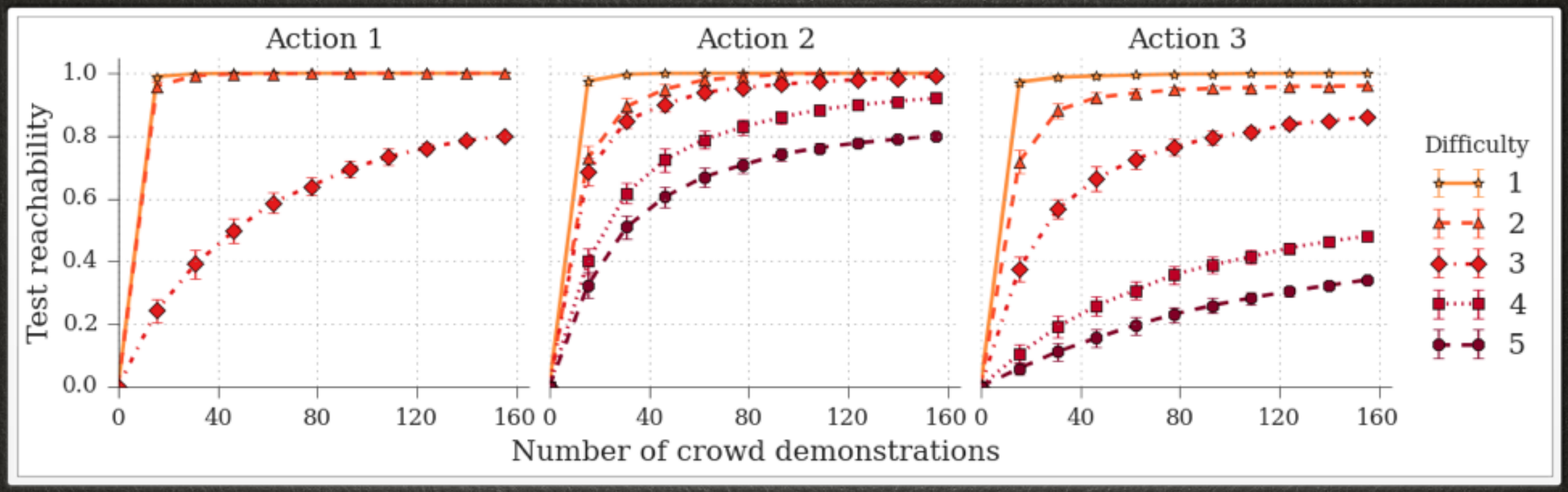
SIDE NOTE ON PICTURES: CROP WELL (PART 2)



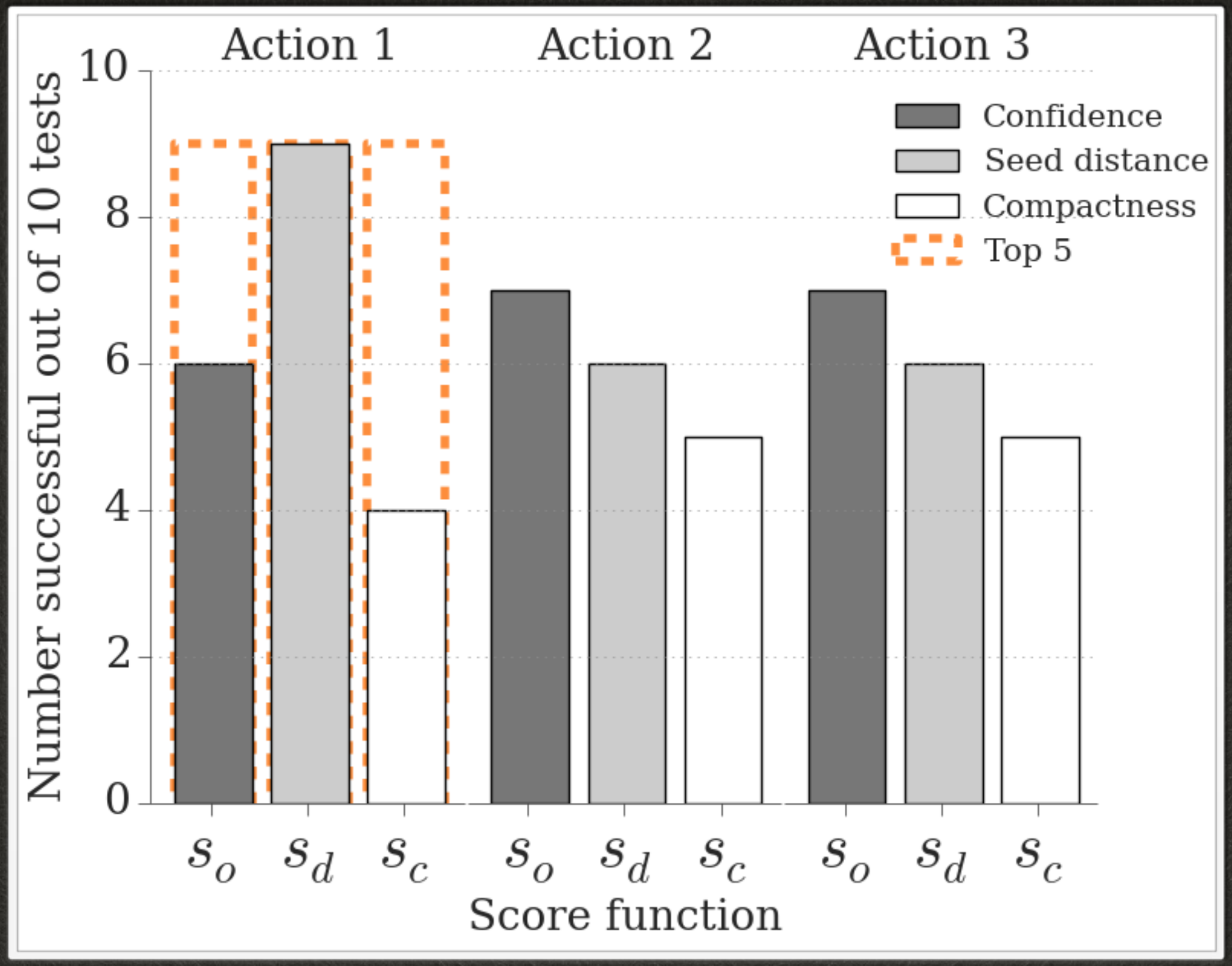
SIDE NOTE ON PICTURES: CROP WELL (PART 2)



SQUEEZE FOR SPACE (PART 1)



SQUEEZE FOR SPACE (PART 2)



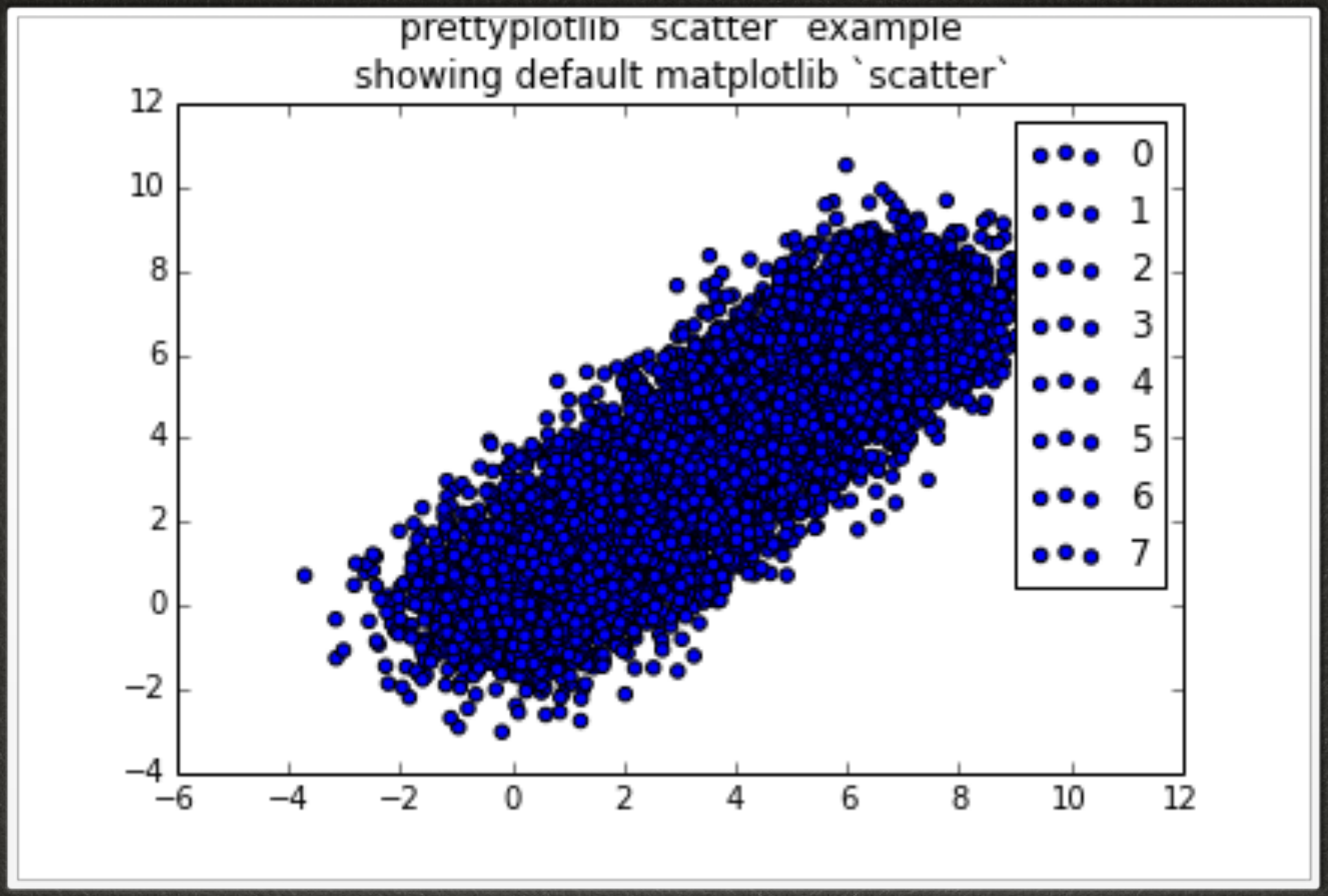
SQUEEZE FOR SPACE (PART 3)

3. CASE STUDIES

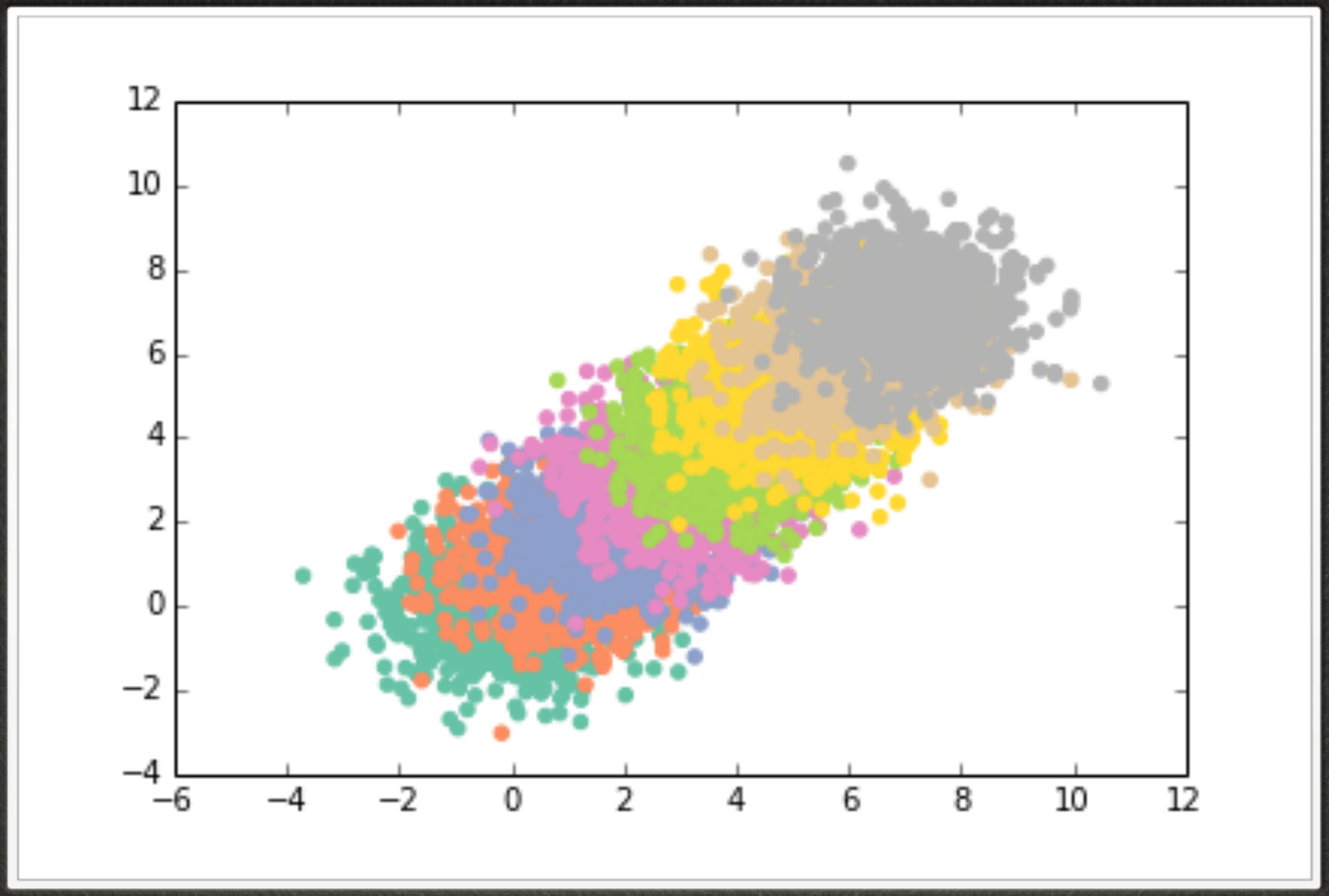
Prettiness, blow-by-blow.

Case study 1: Olga's Scatter

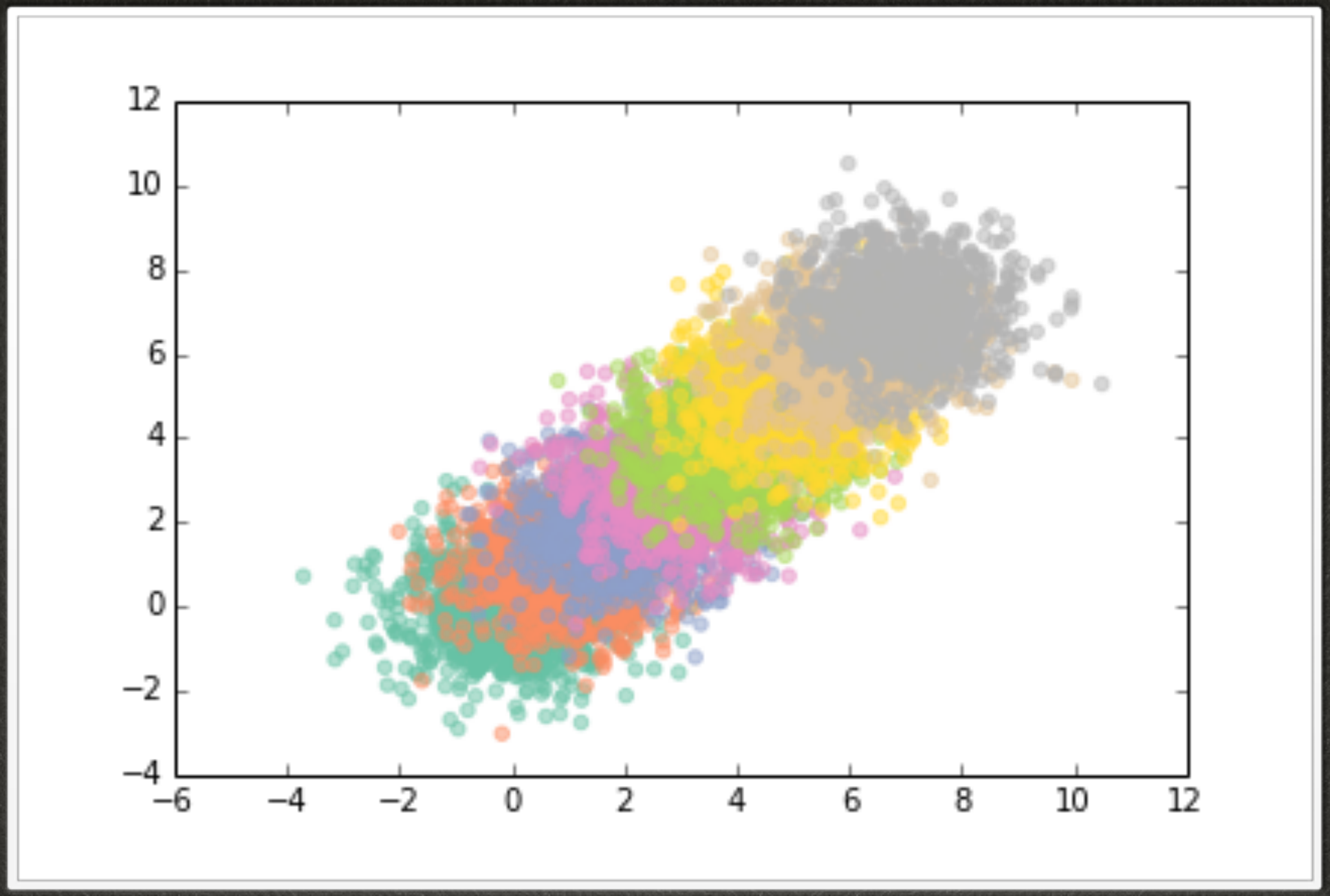
"'Dis ain't no uglyplotlib' - Anonymous" - Olga Botvinnik


prettyplotlib scatter example
showing default matplotlib `scatter`
12
10
8
6
4
2
0
−2
−4
−6
−4
−2
0
2
4
6
8
10
12
0
1
2
3
4
5
6
7

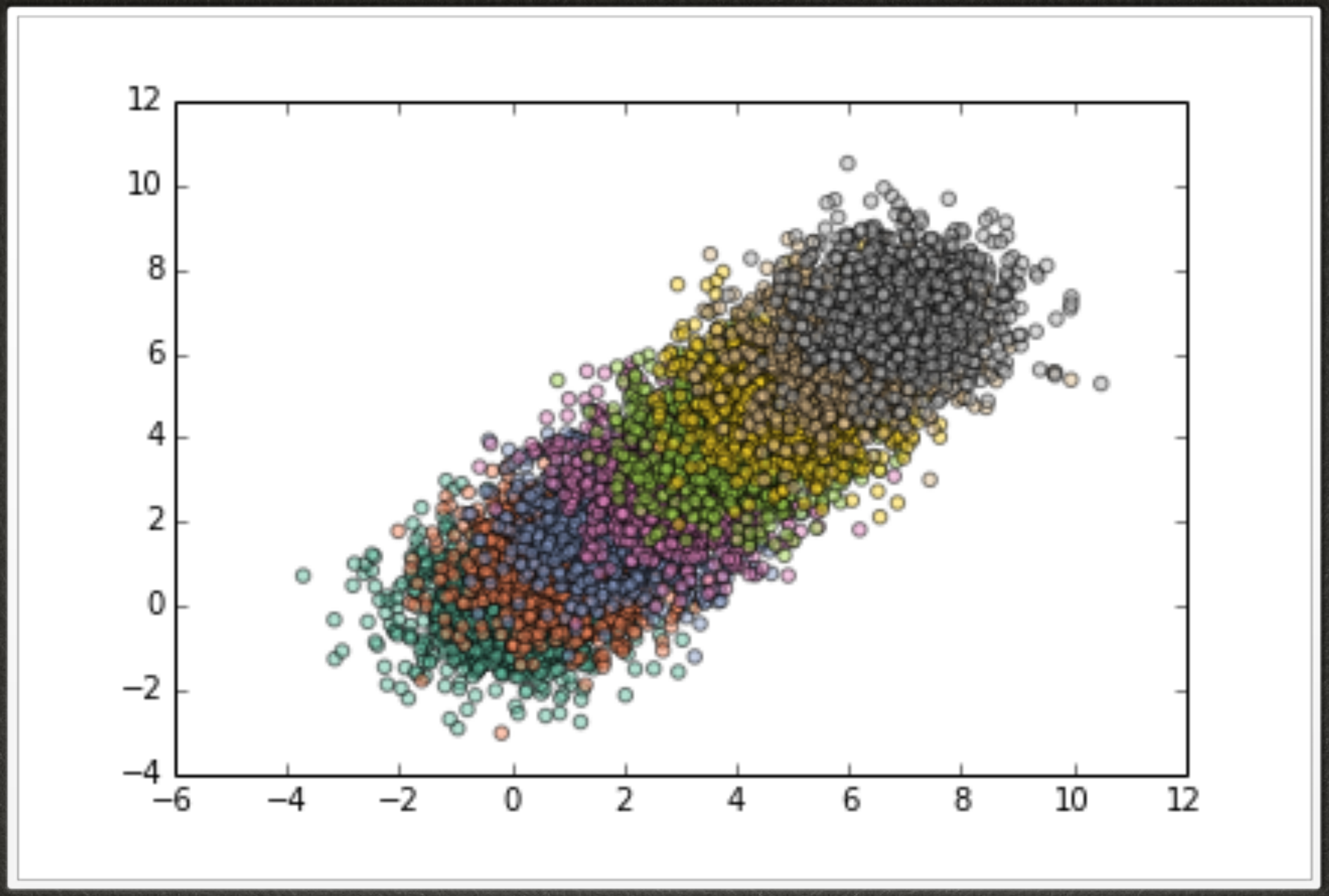
1: DEFAULT



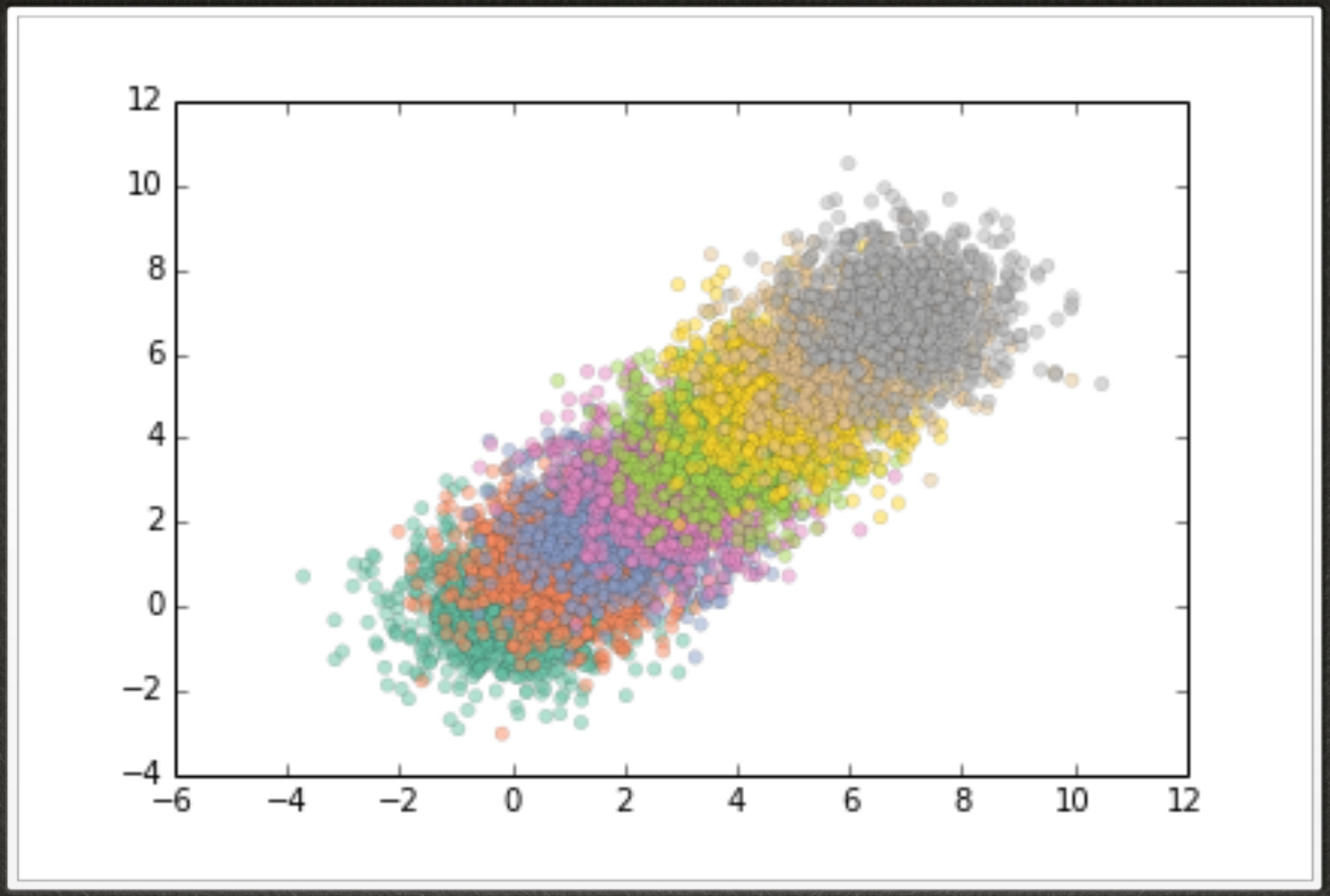
2: USE BETTER COLORS



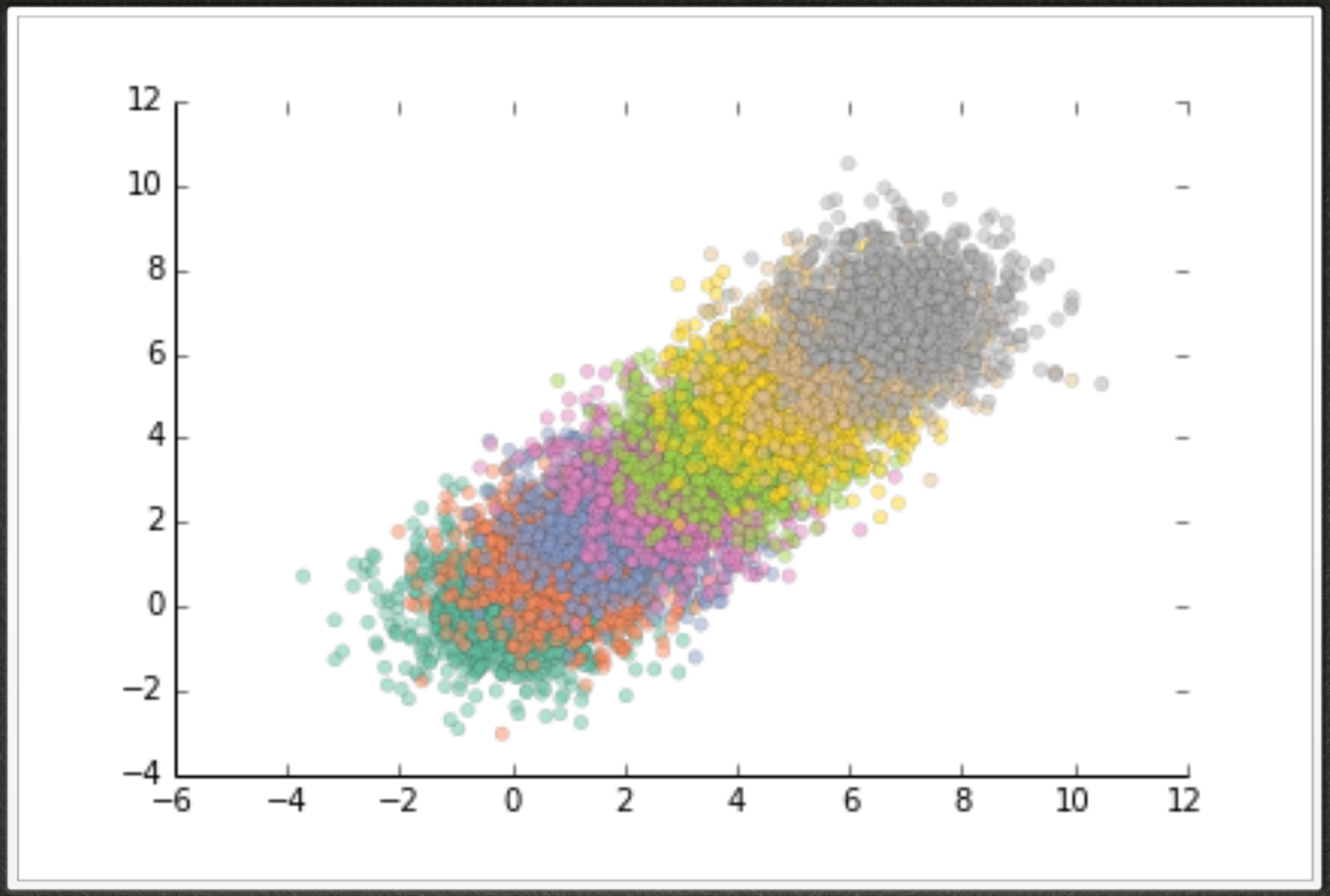
3: ADD ALPHA (OVERLAP)



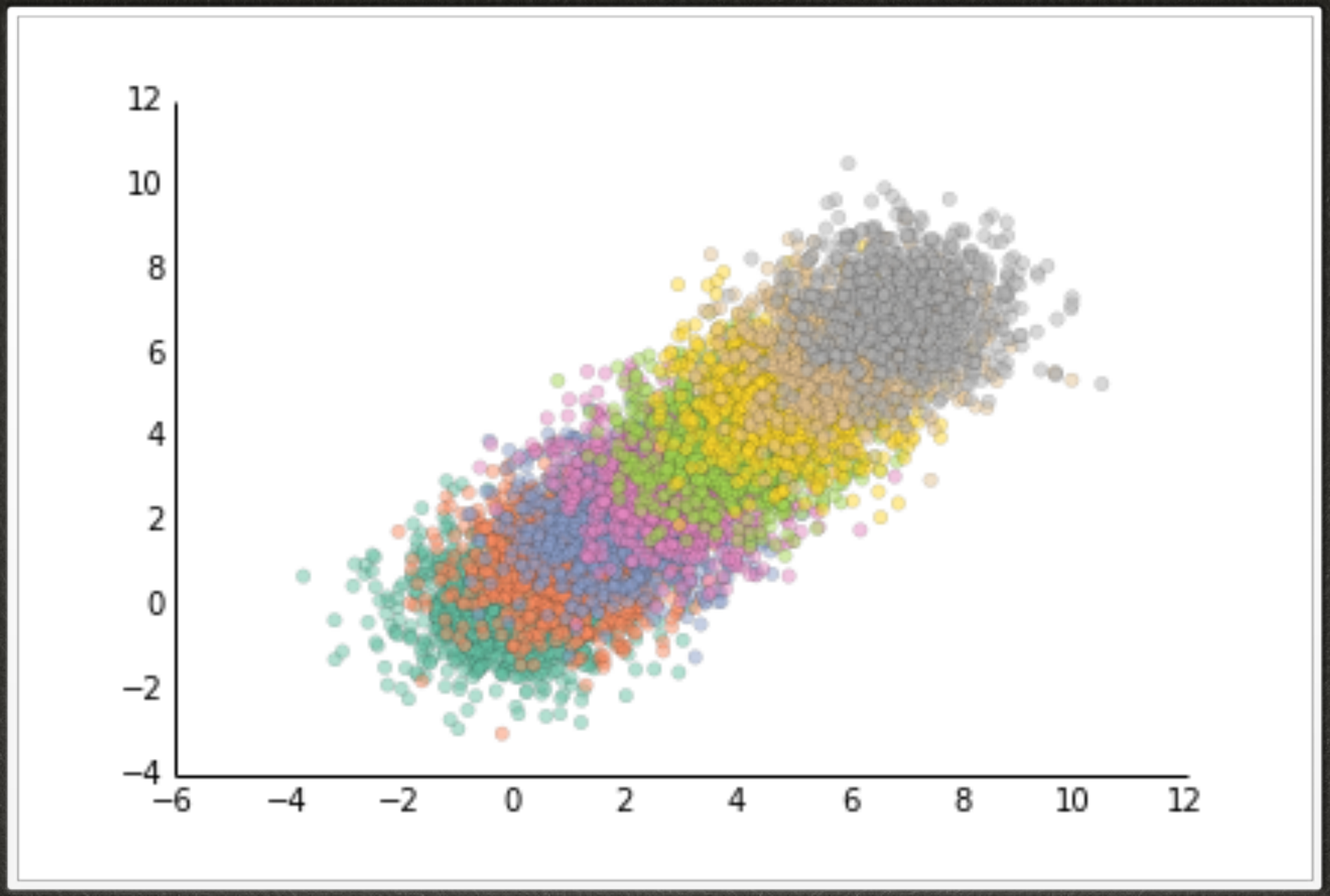
4: ADD OUTLINE



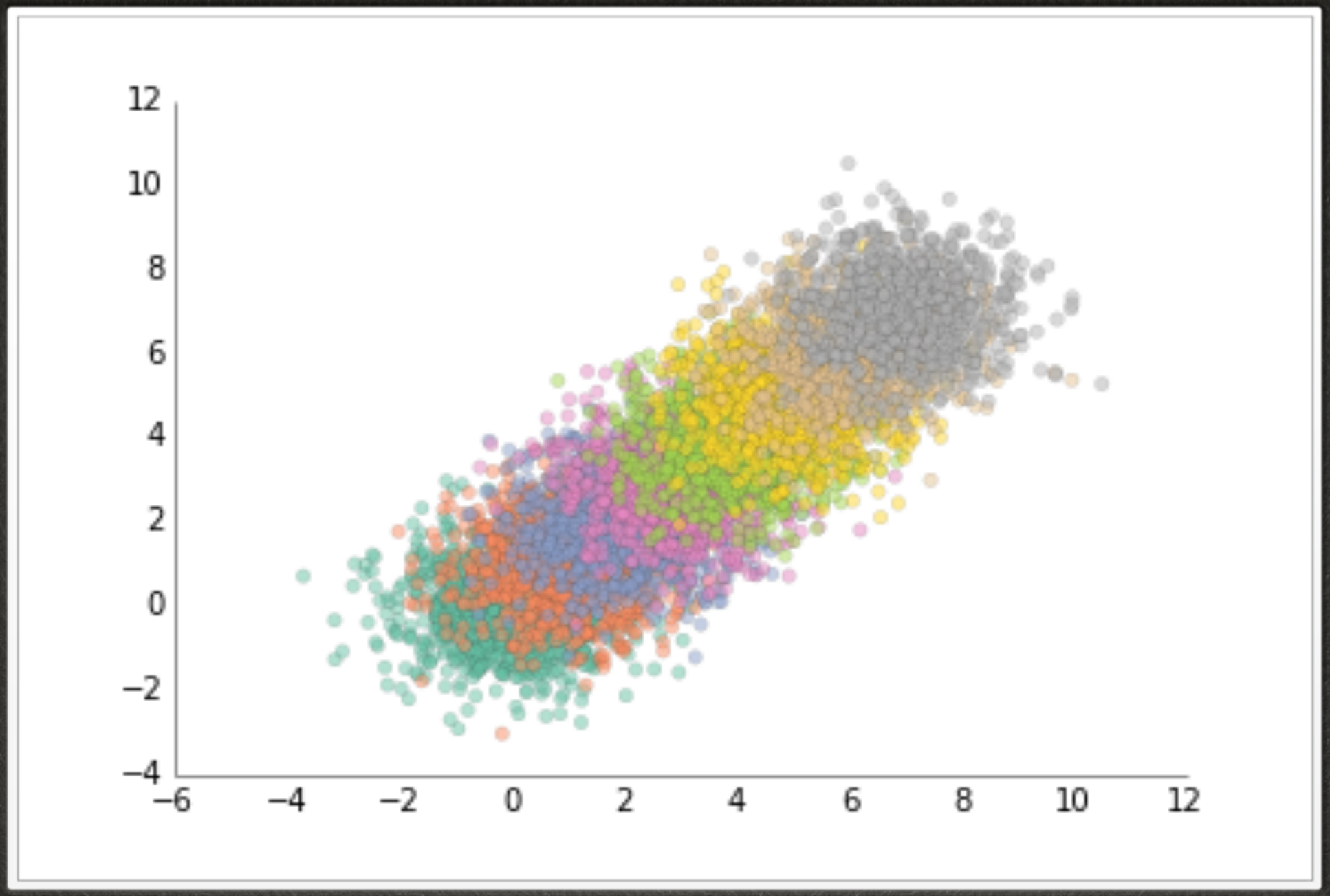
5: THIN THE OUTLINE



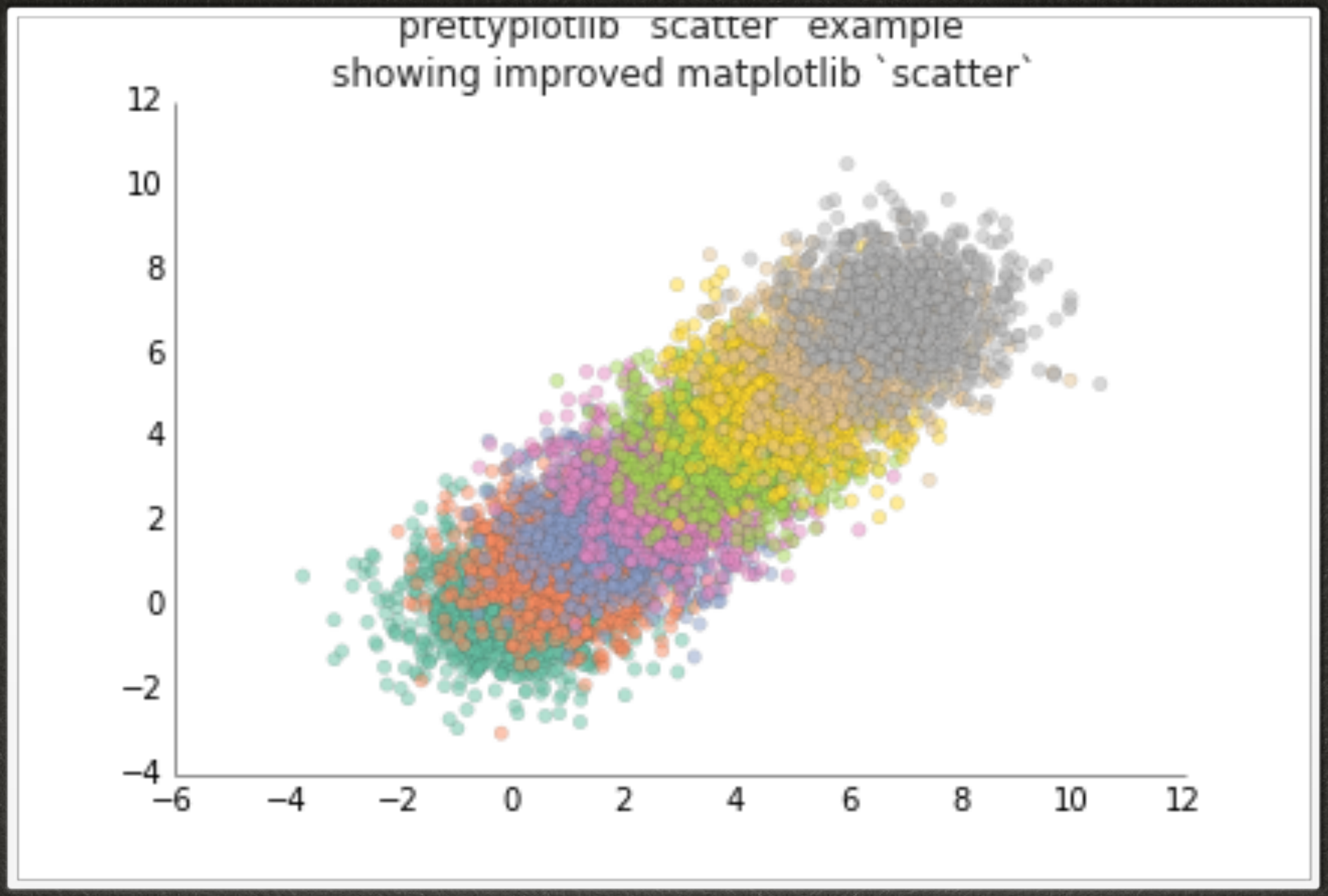
6: REMOVE T&R SPINES



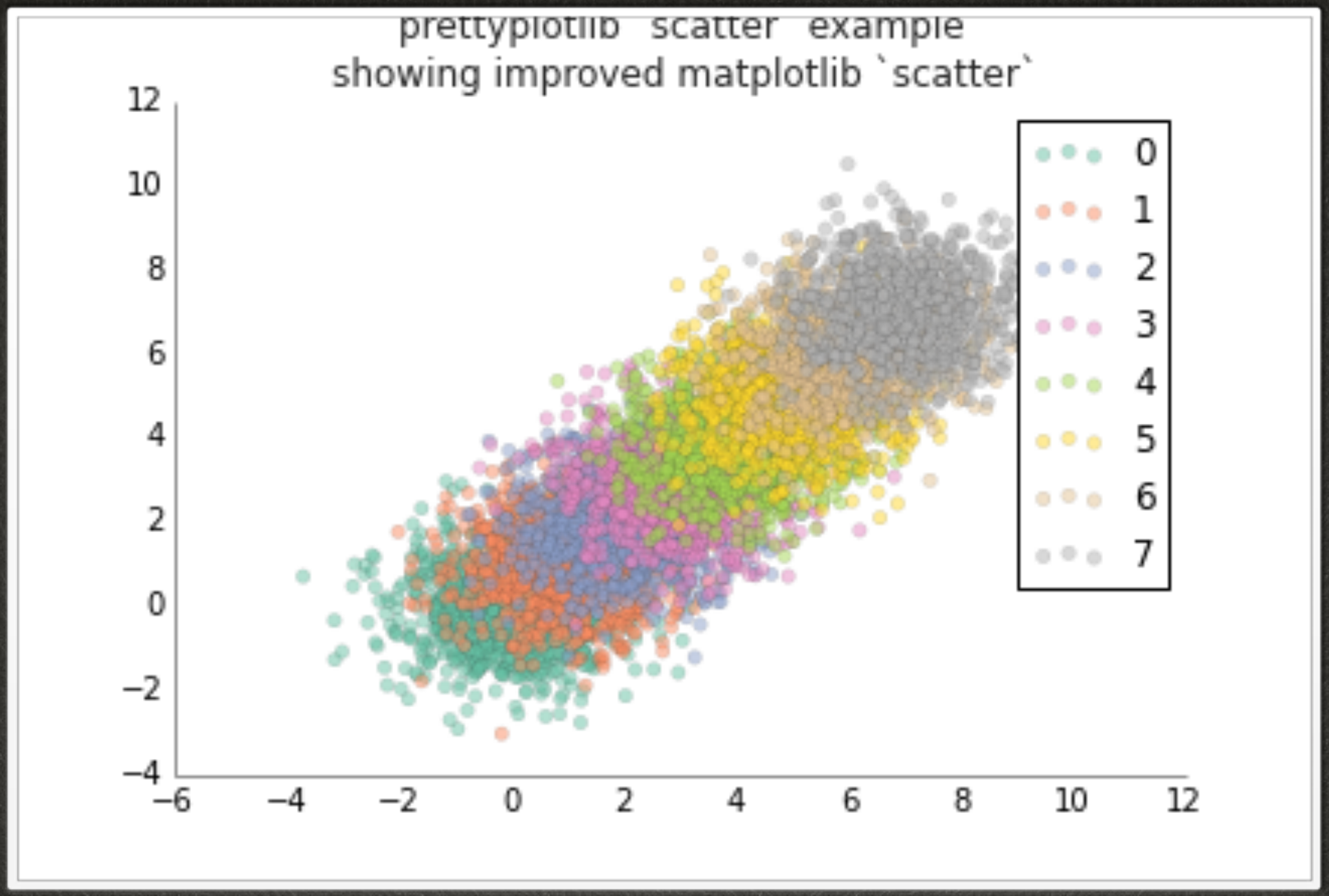
7: REMOVE ALL TICKS



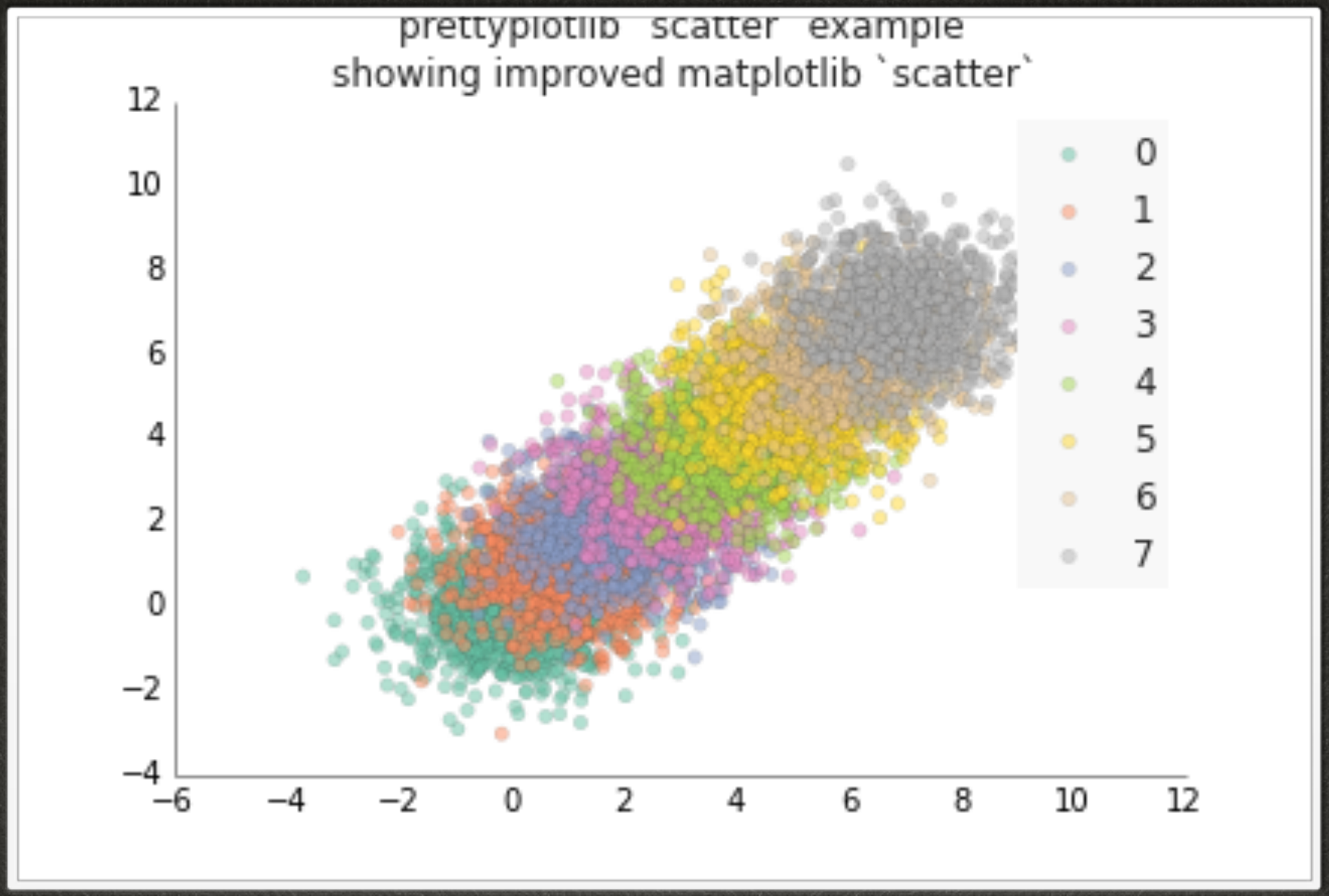
8: MAKE SPINES OFF-BLACK



9: MAKE TICK LABELS OFF-BLACK

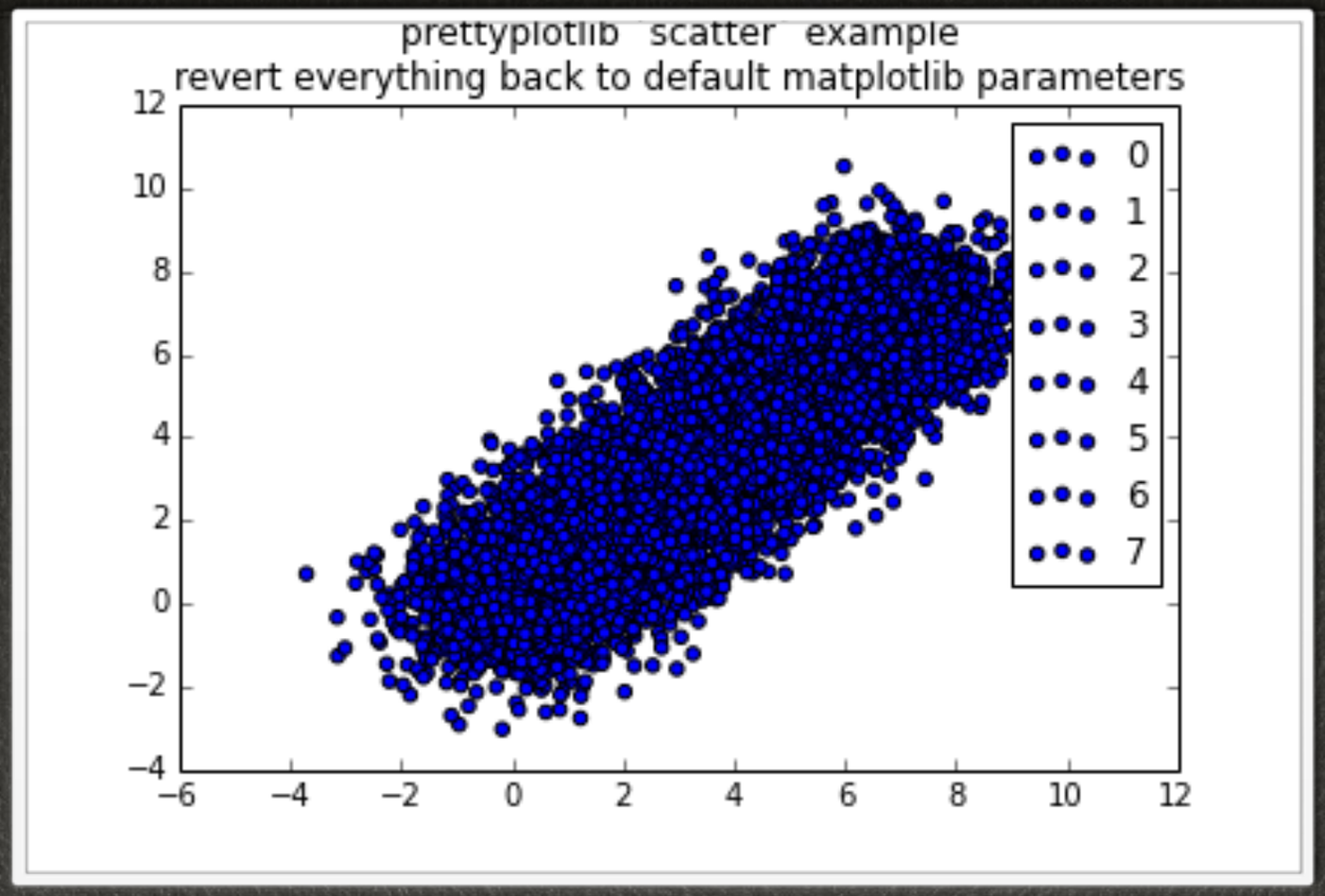


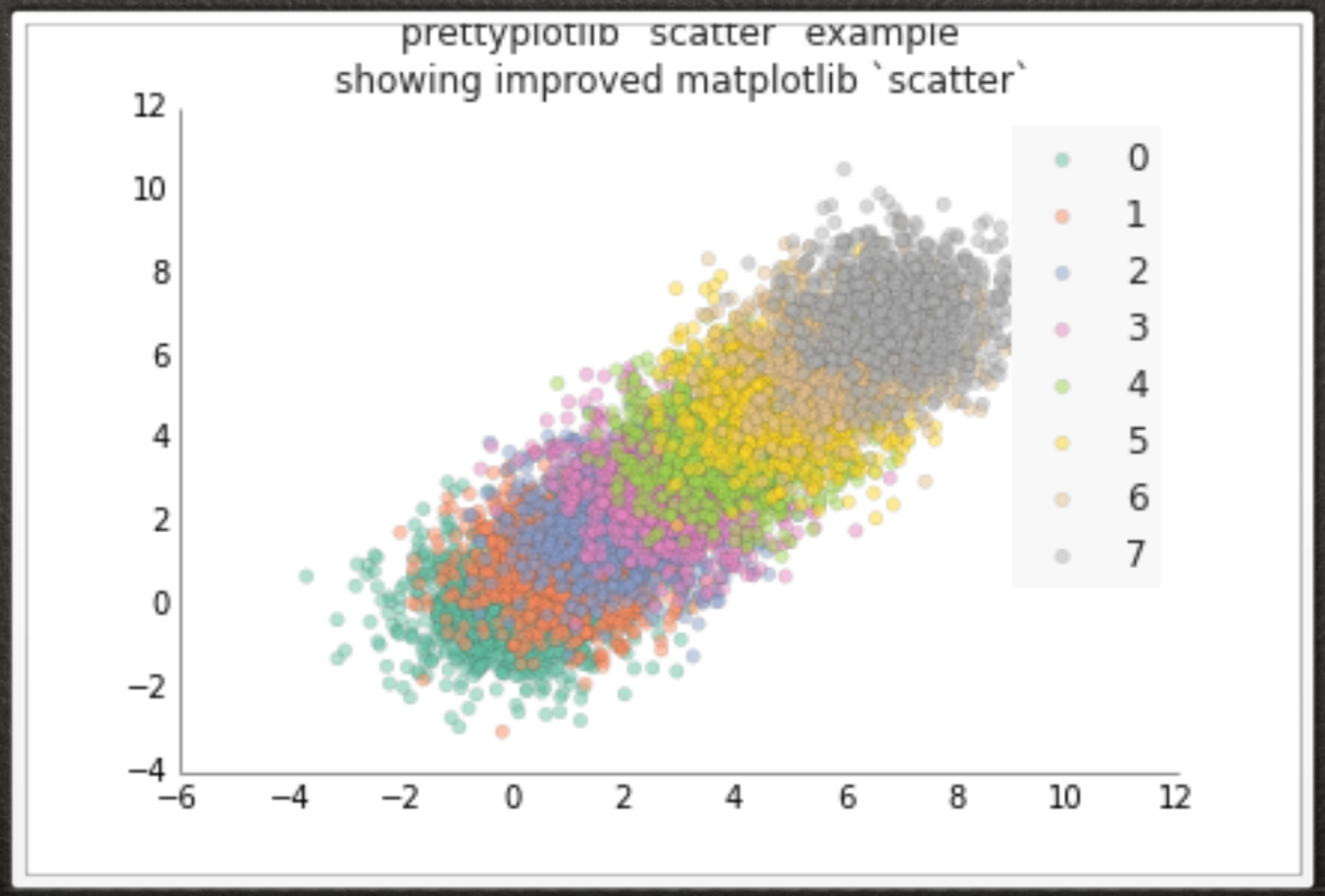
10: ADD LEGEND



11: LEGEND IMPROVEMENTS

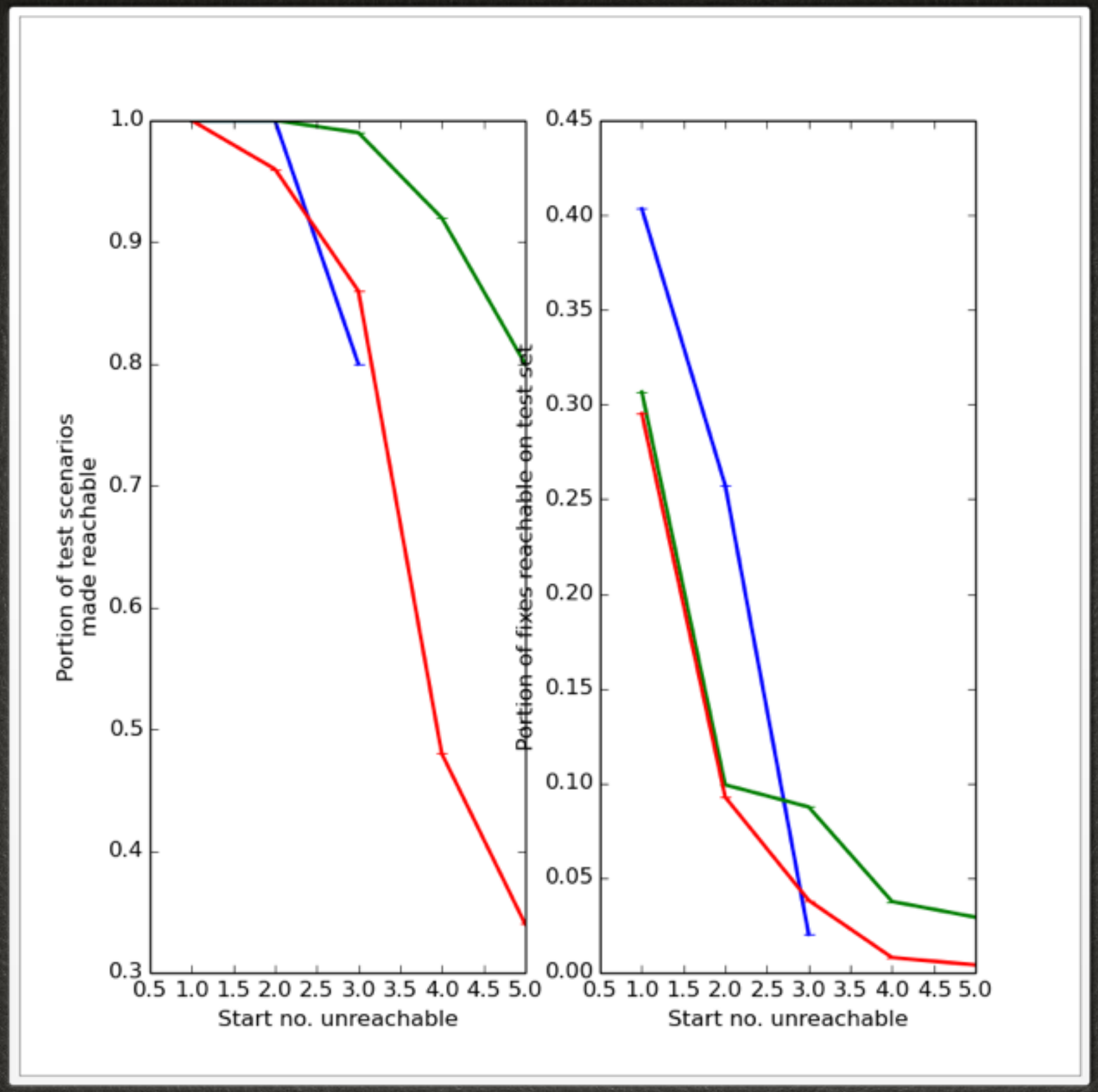
FINAL COMPARISON



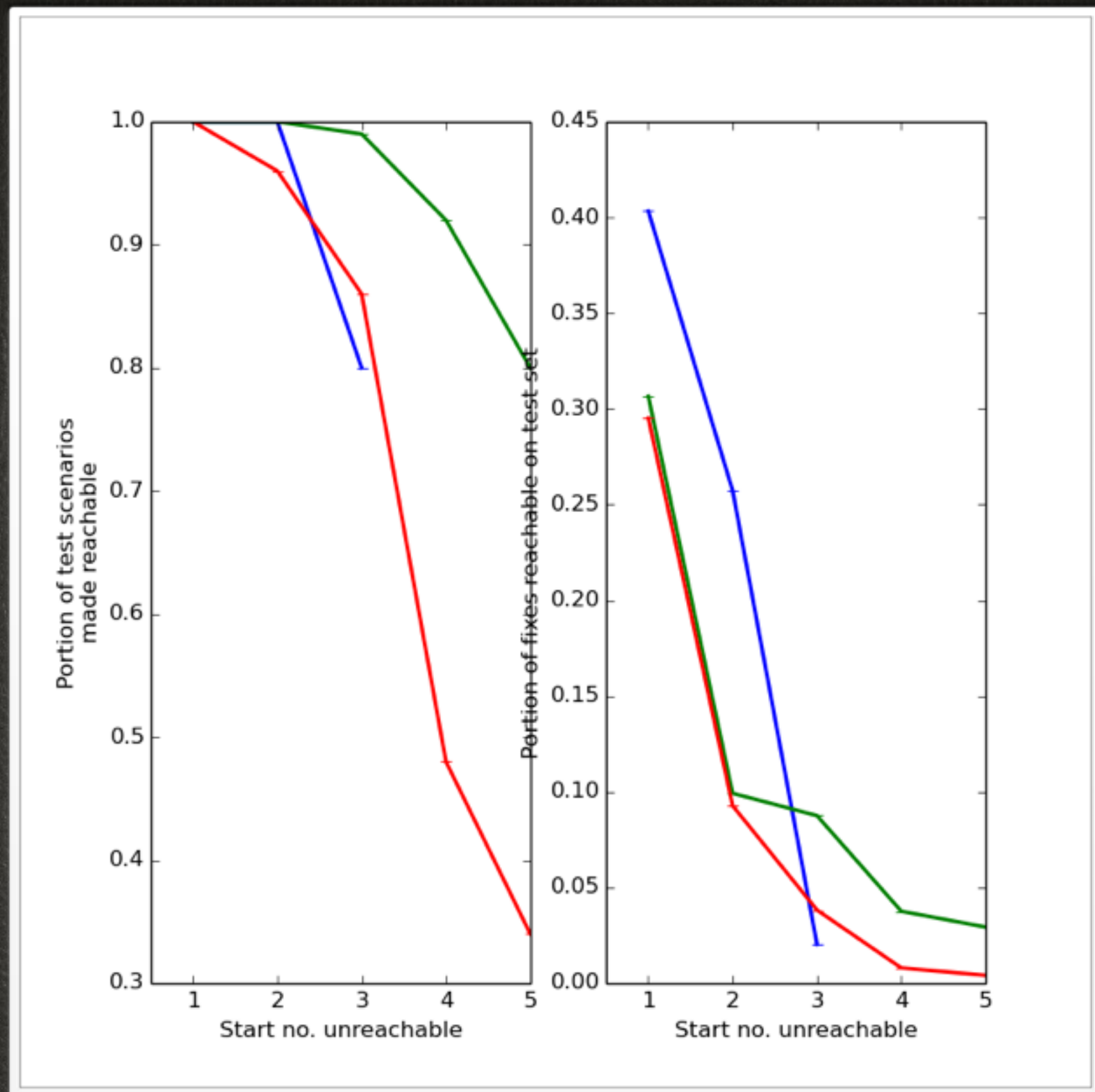


Case study 2: Double-graph for HCOMP paper

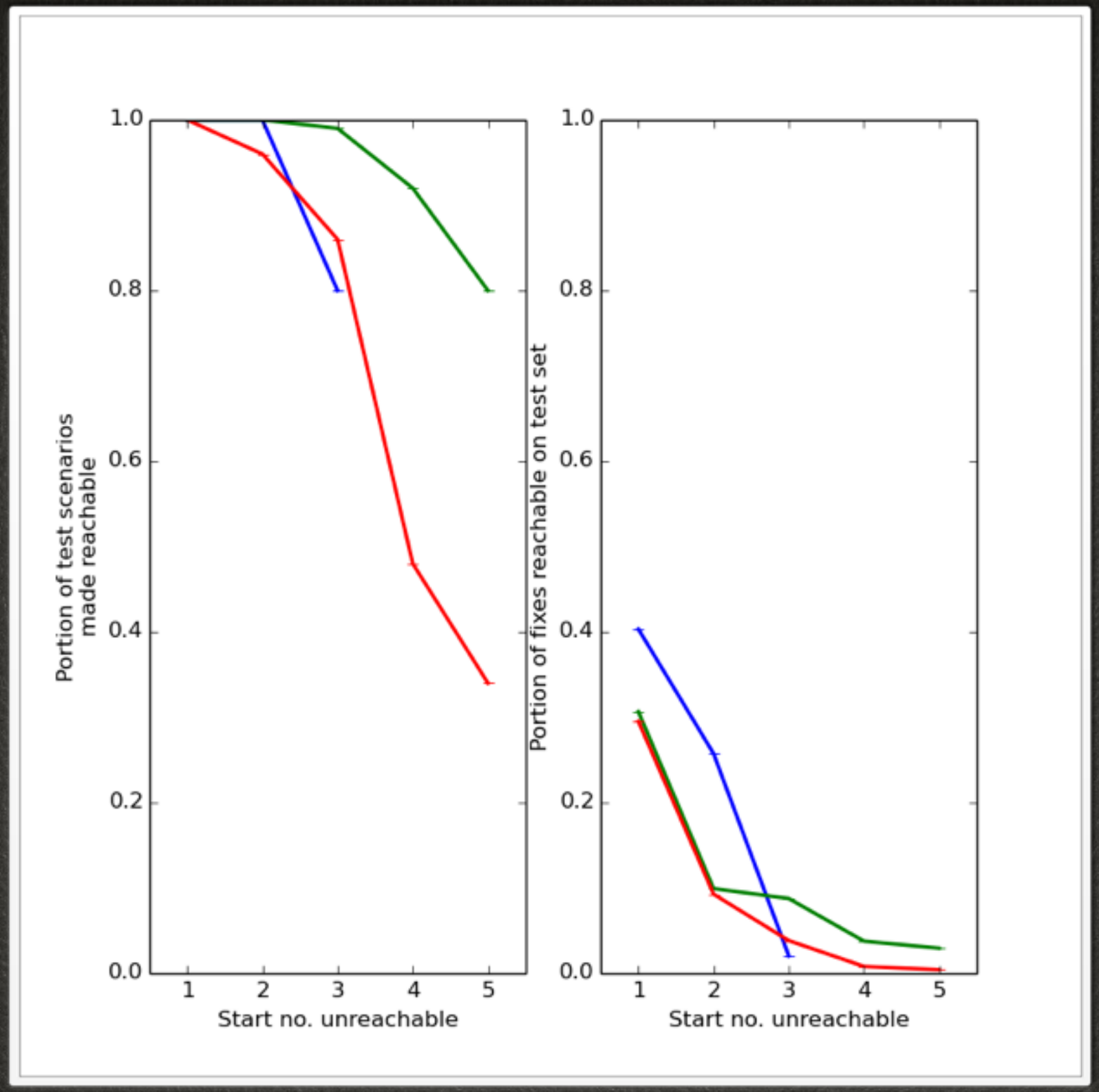
“I really have no idea what I’m doing” - me


Portion of test scenarios made reachable
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.5 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0
Start no. unreachable
Portion of fixes reachable on test set
0.45
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
0.5 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0
Start no. unreachable

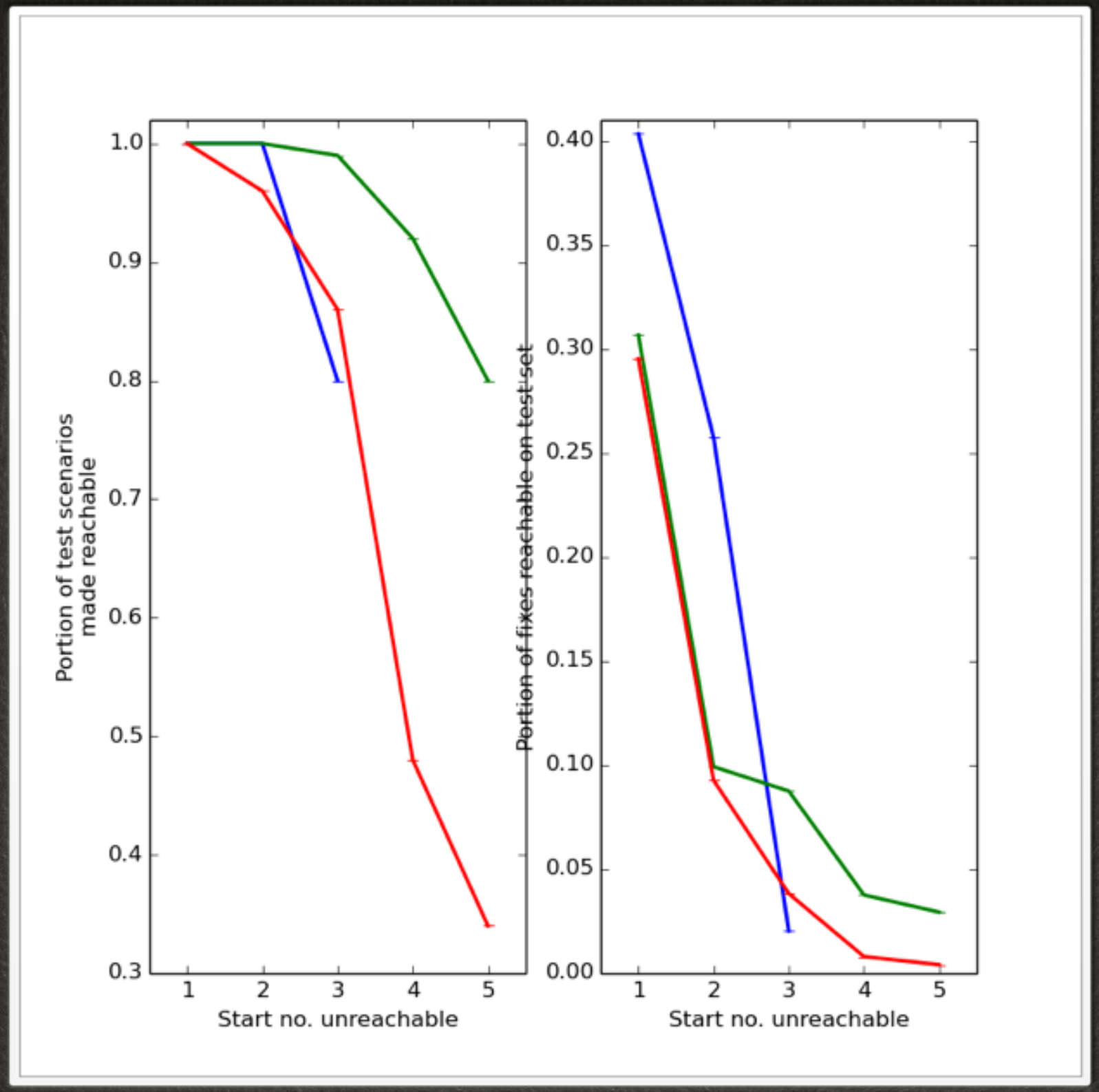
1: START



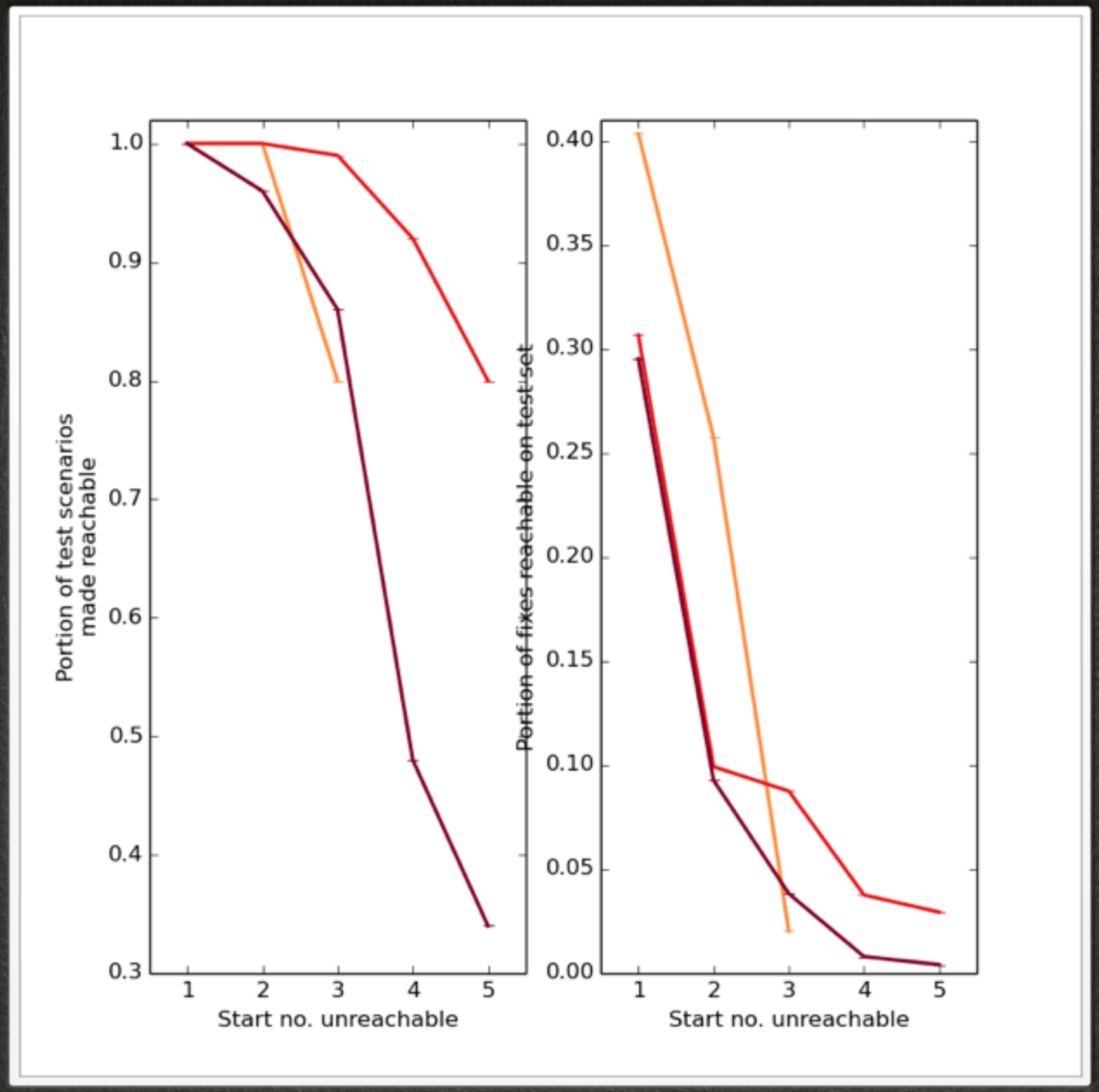
2: CHANGE X-AXIS LABELS



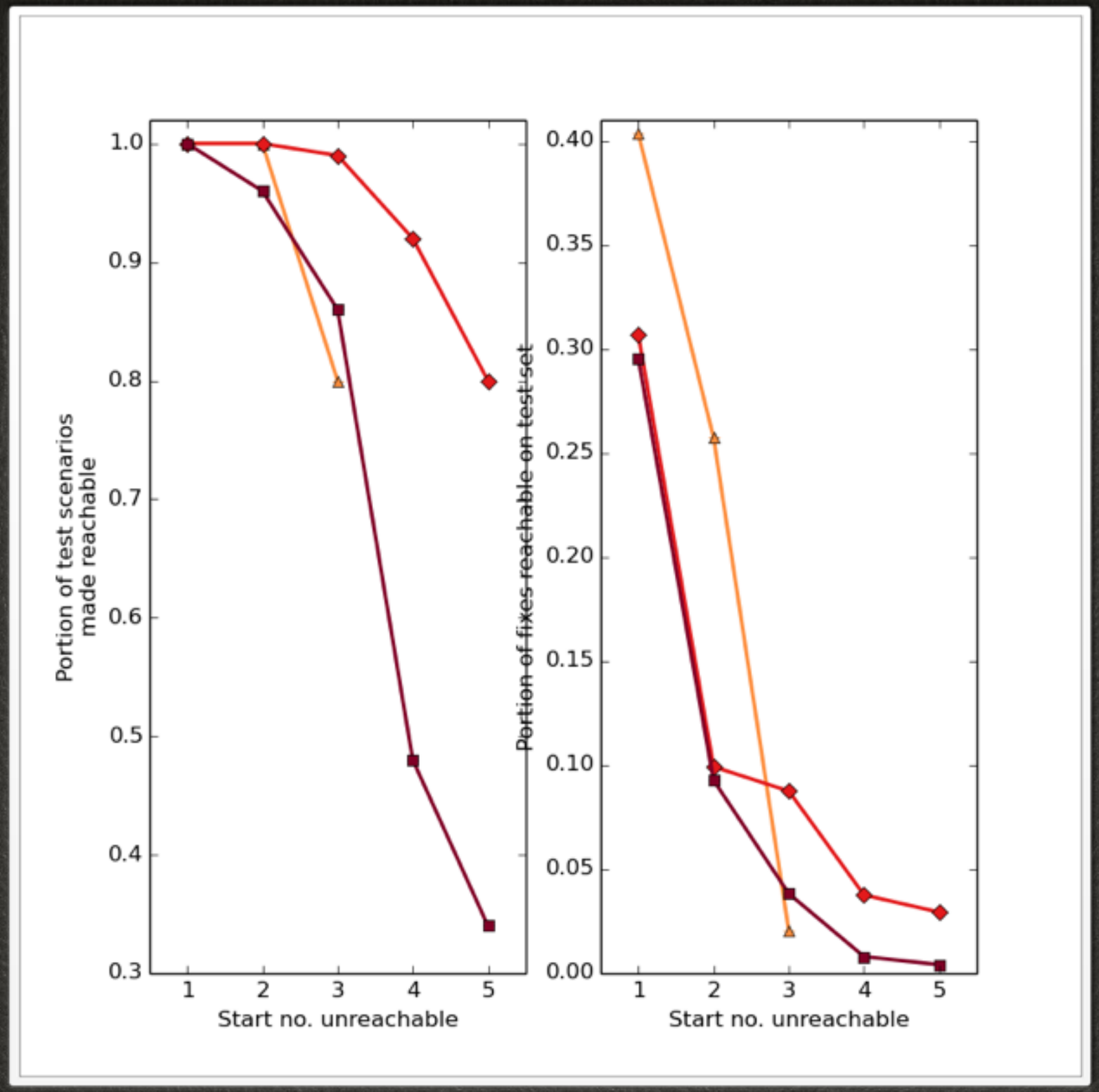
3: CHANGE X-AXIS RANGE



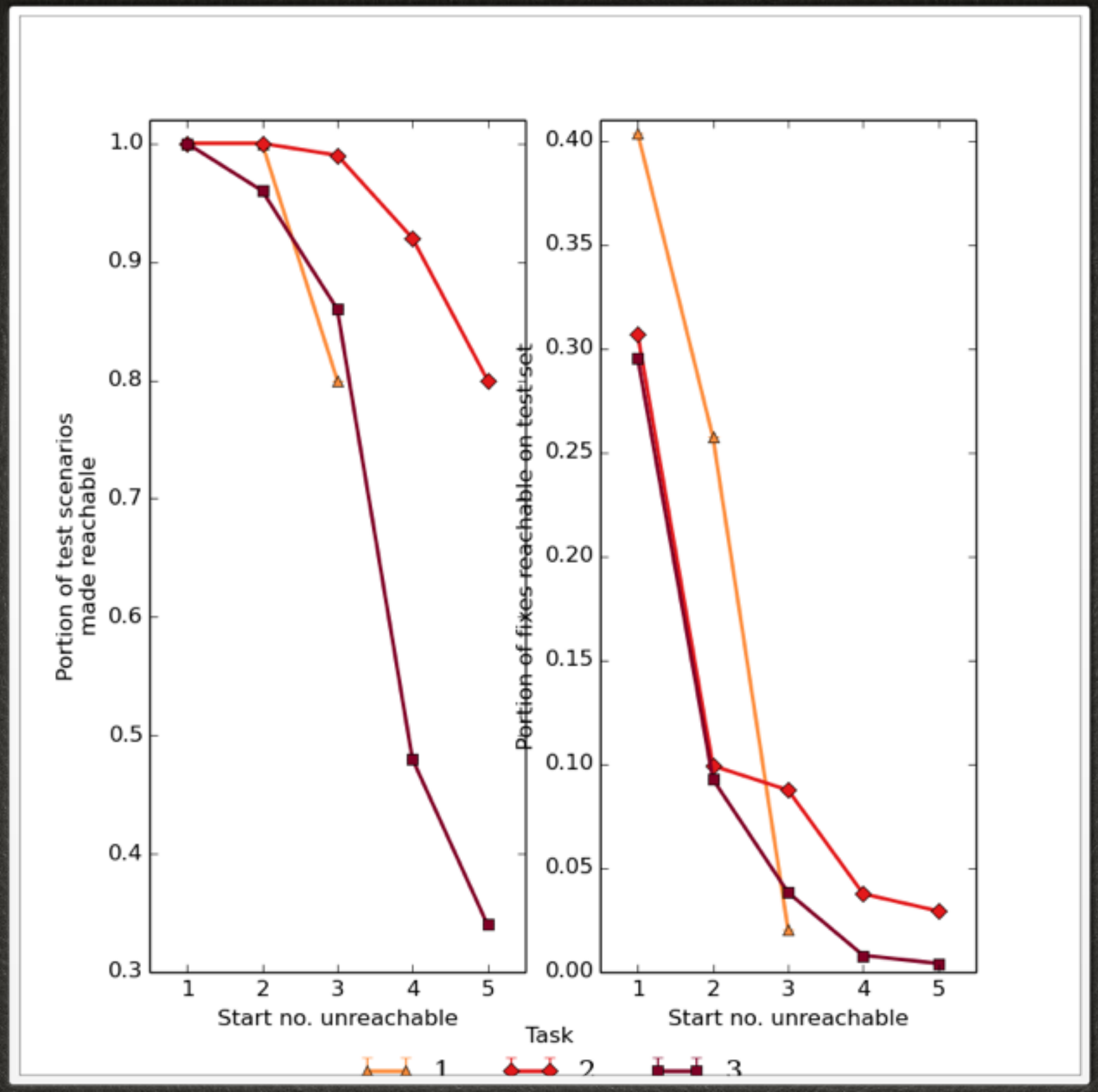
4: CHANGE Y-AXIS RANGE



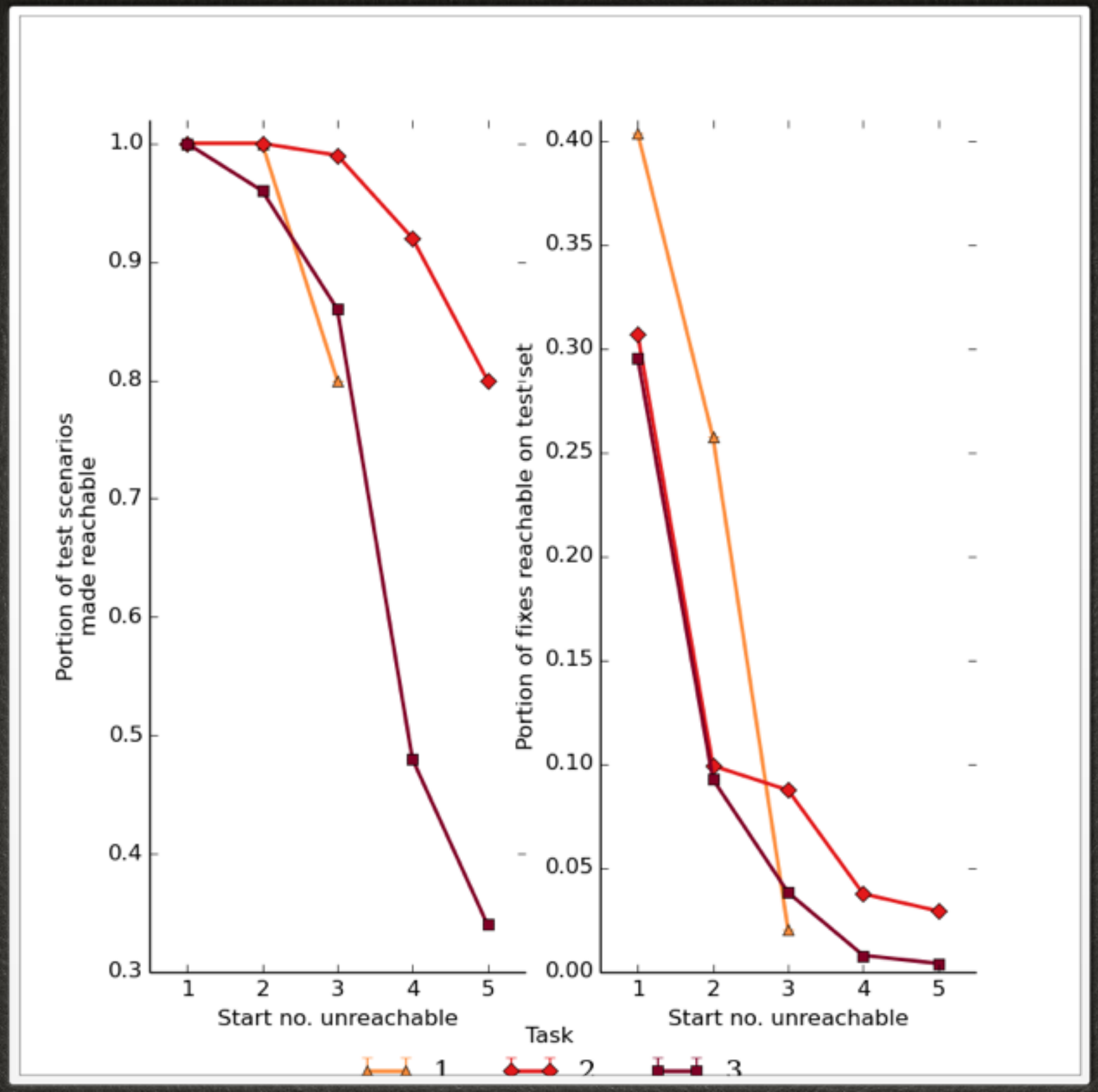
5: CHANGE COLORS



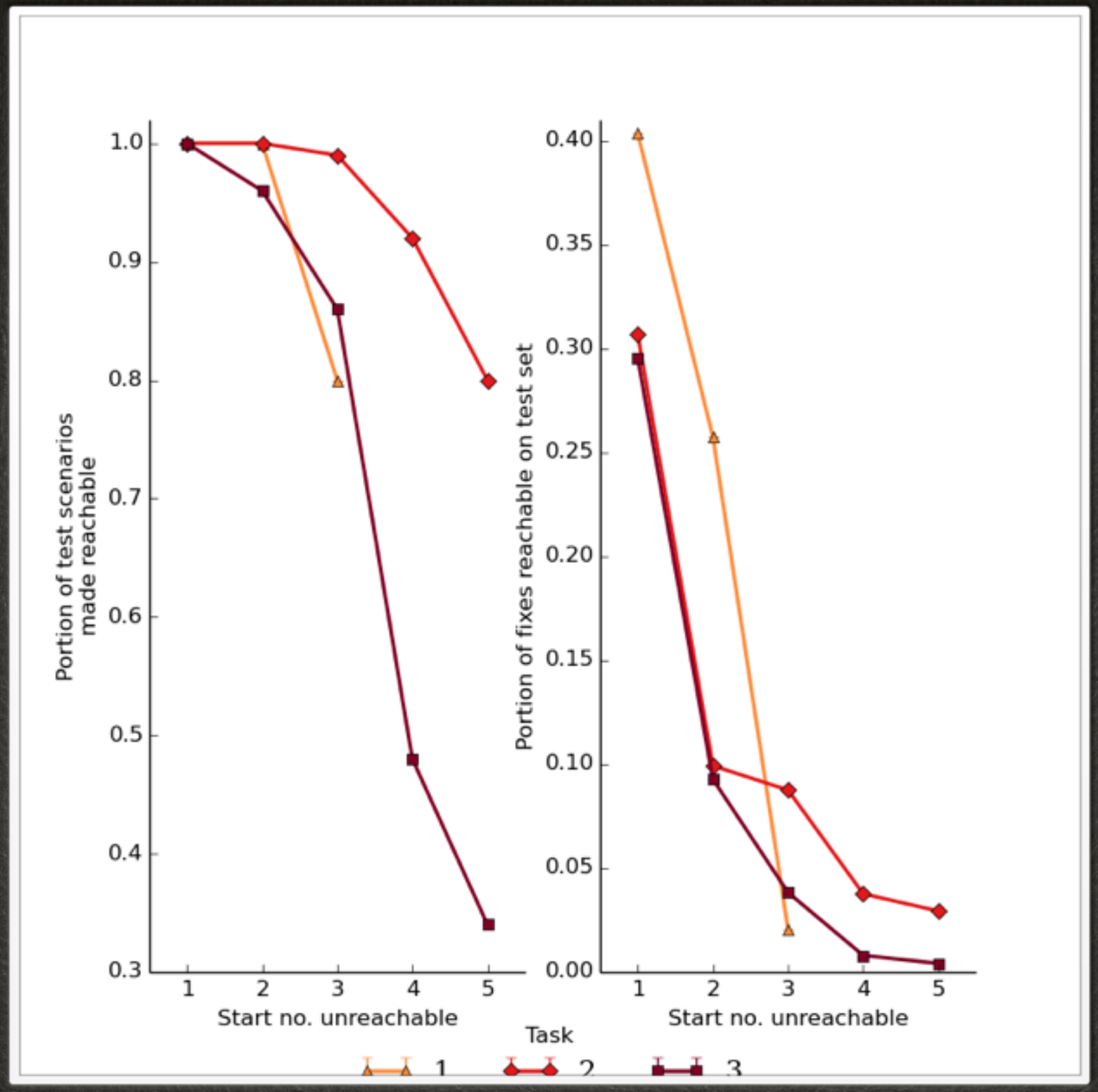
6: ADD MARKERS


Portion of test scenarios made reachable
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
1
2
3
4
5
Start no. unreachable
Portion of fixes reachable on test set
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
1
2
3
4
5
Start no. unreachable
Task
1
2
3

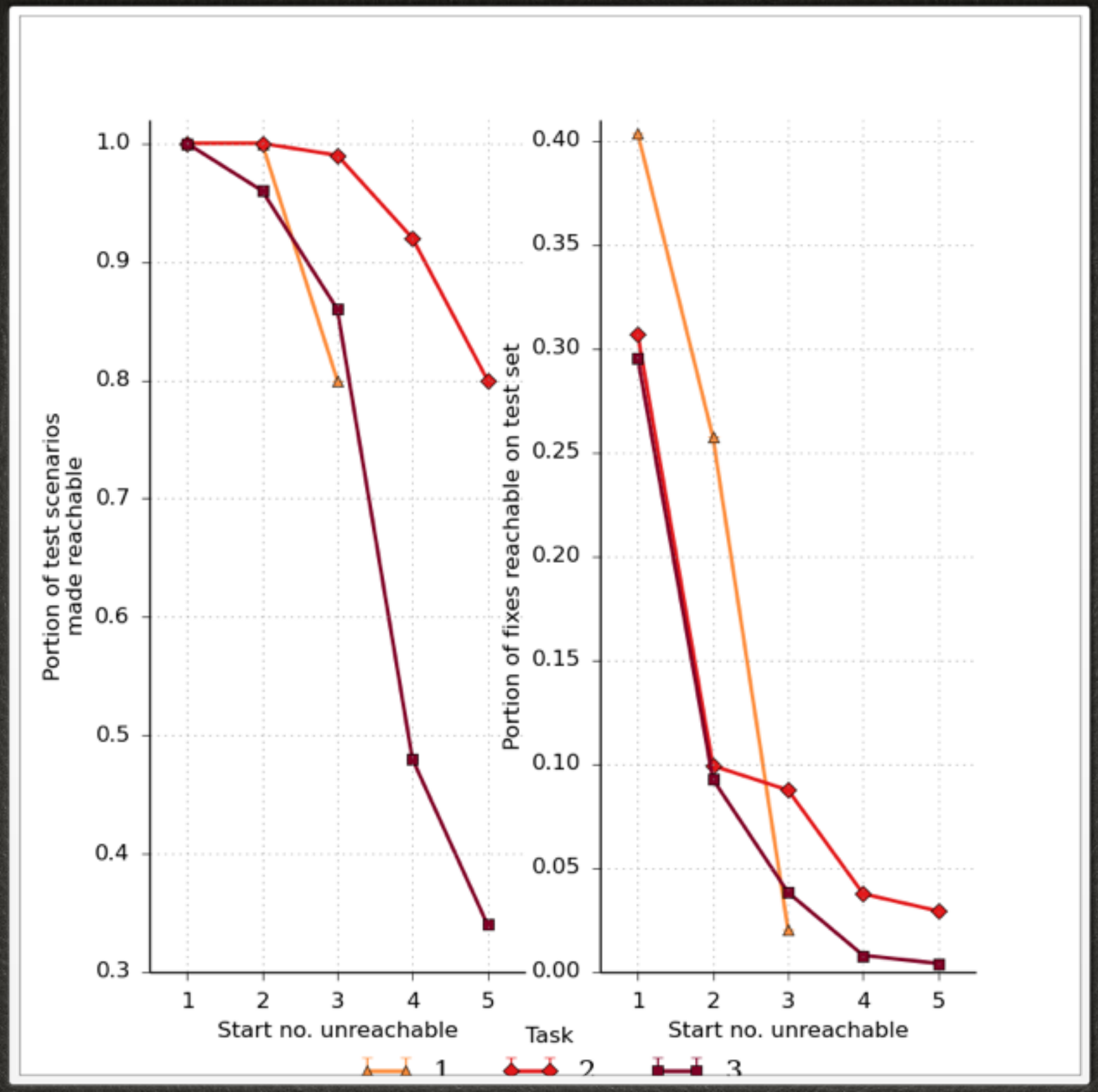
7: ADD LEGEND



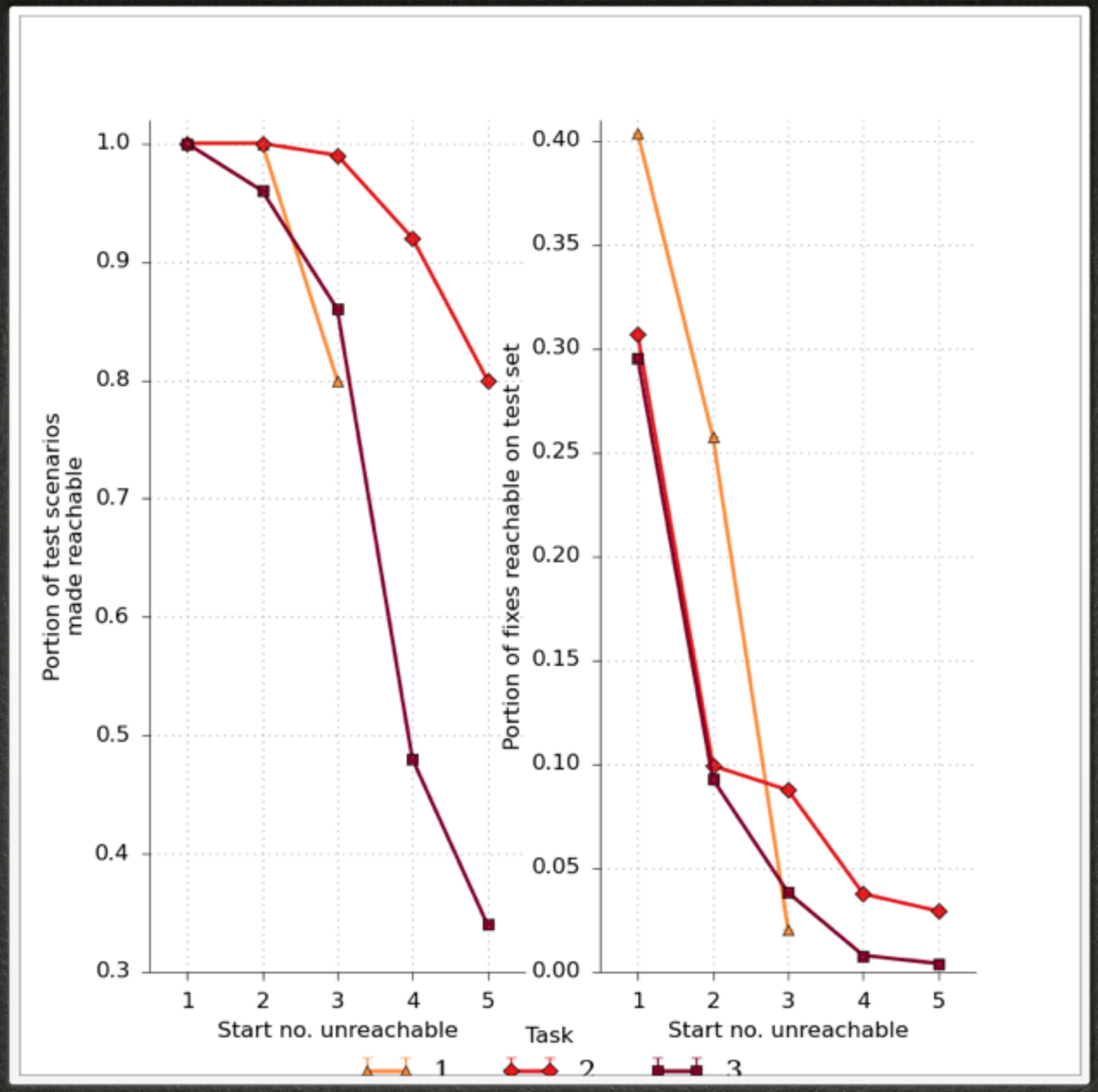
8: REMOVE T&R SPINES



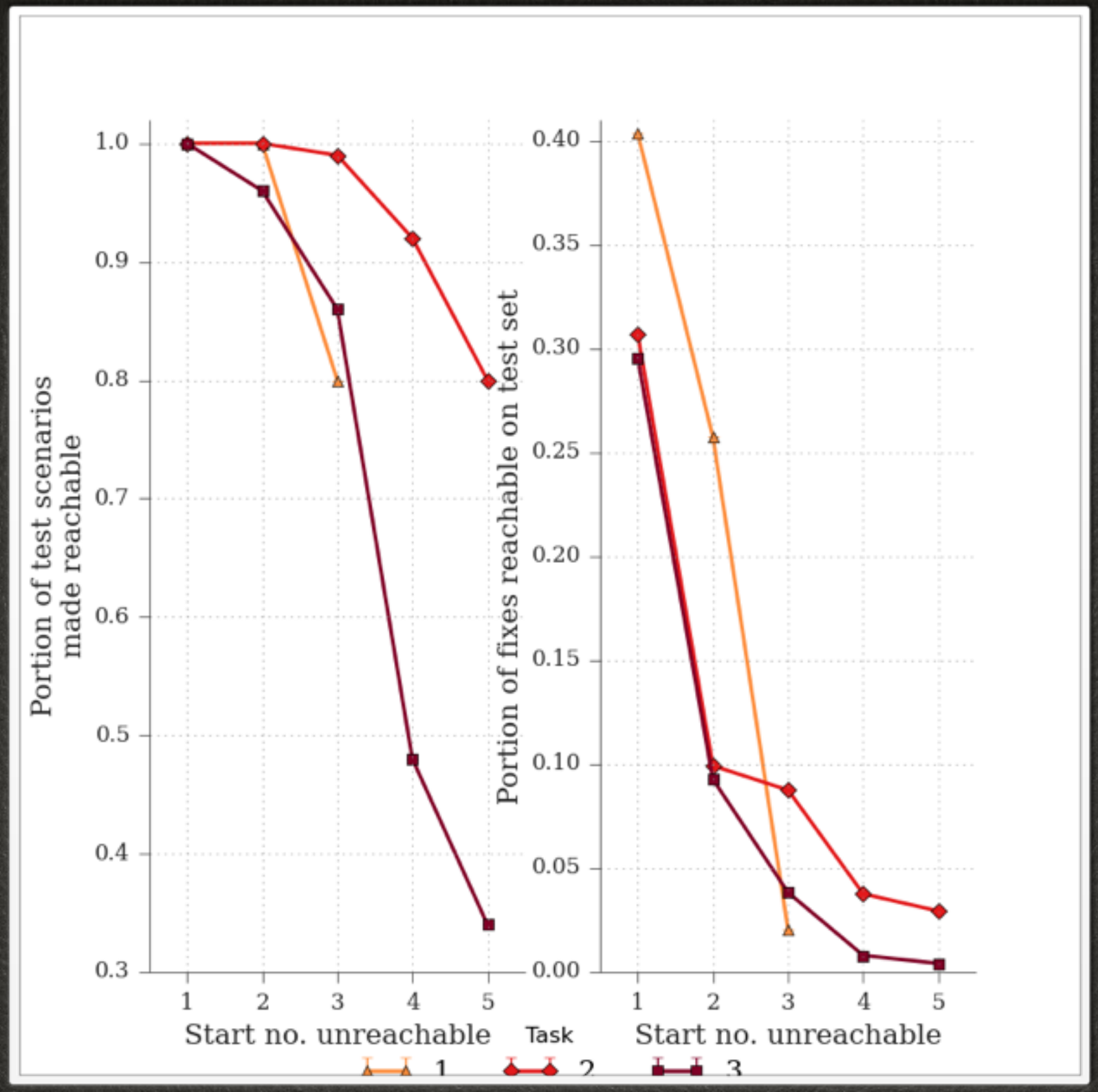
9: REMOVE T&R TICKS



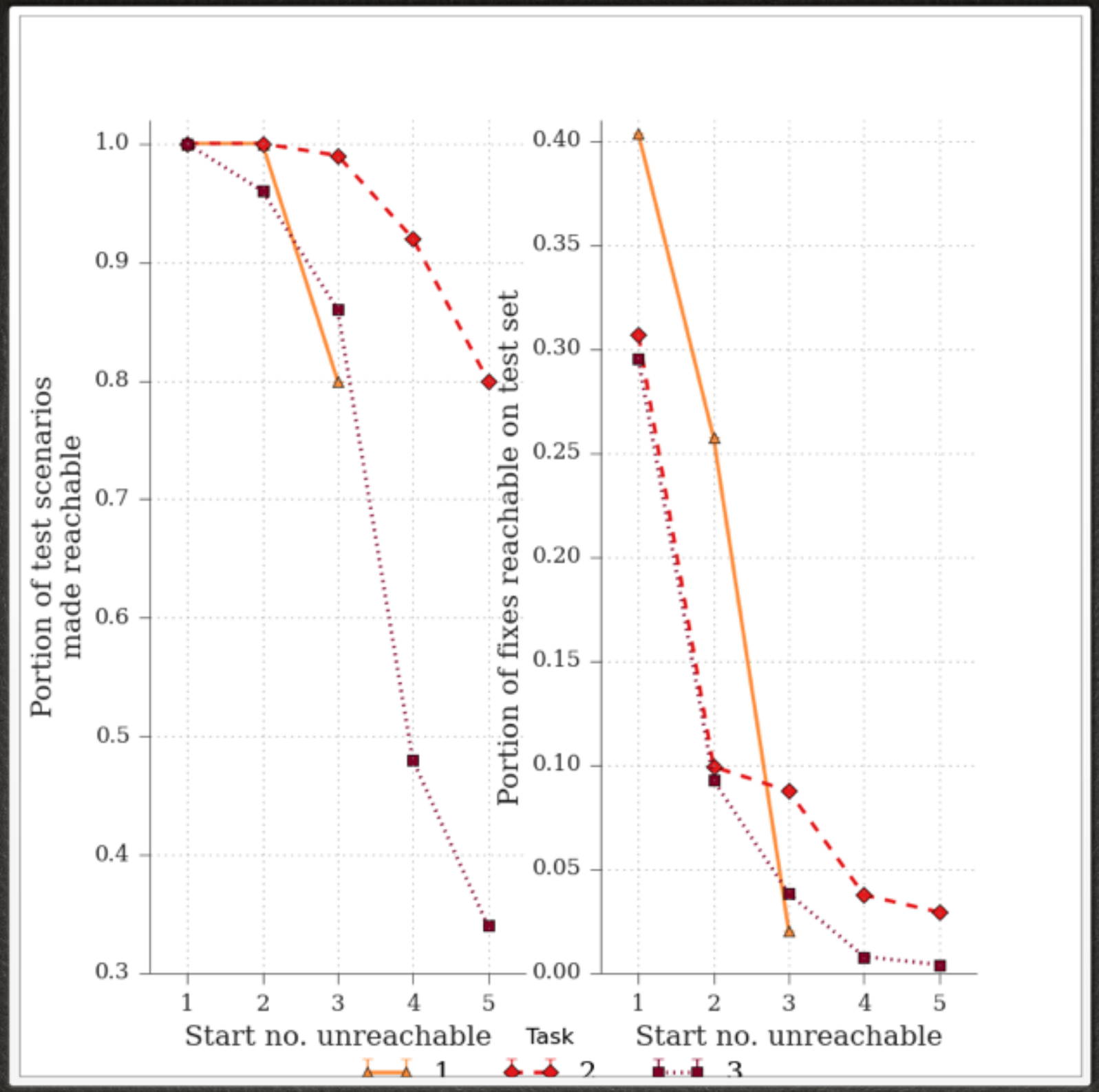
10: TICKS OUT, GRID ON



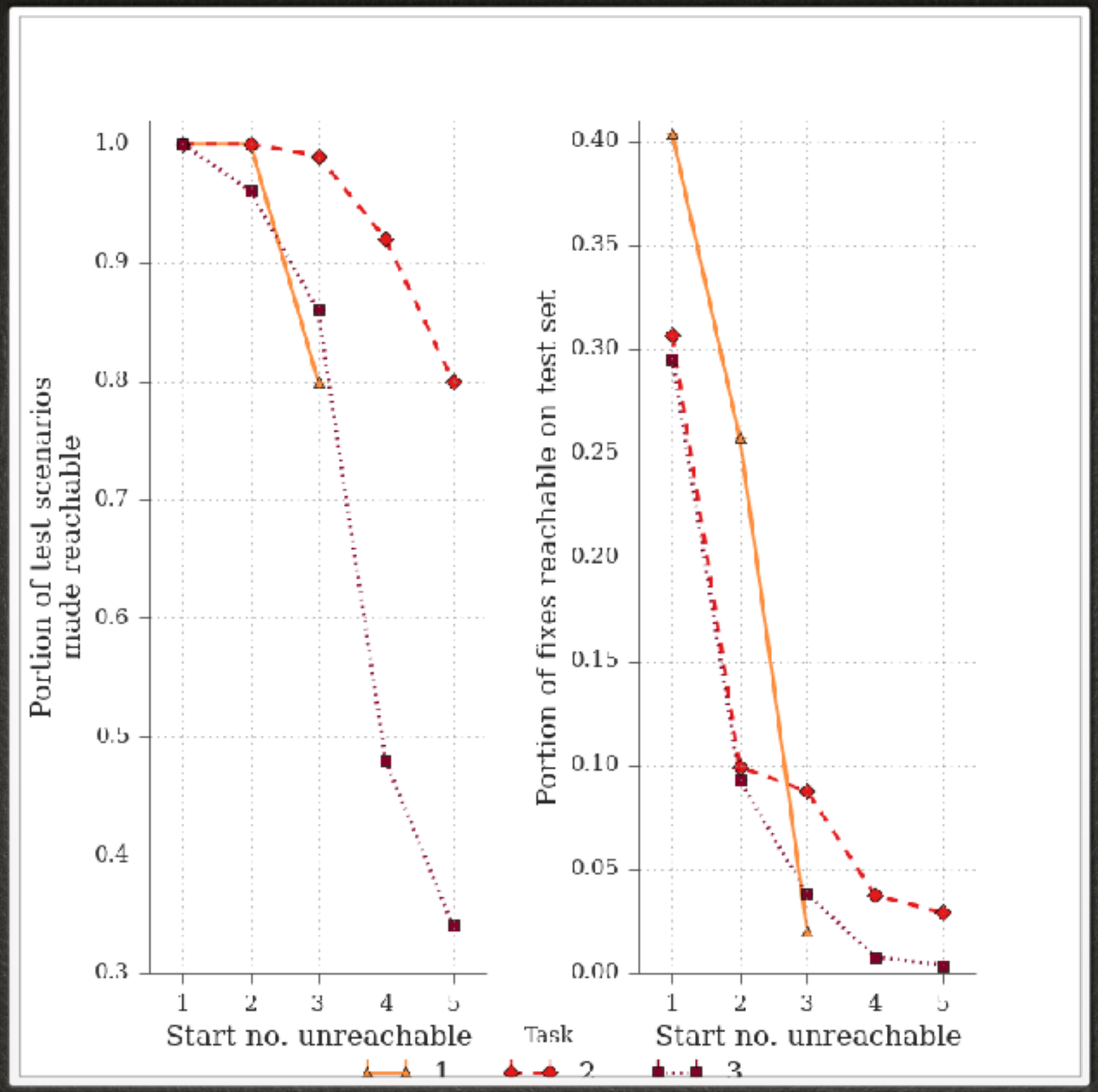
11: SPINES THINNER



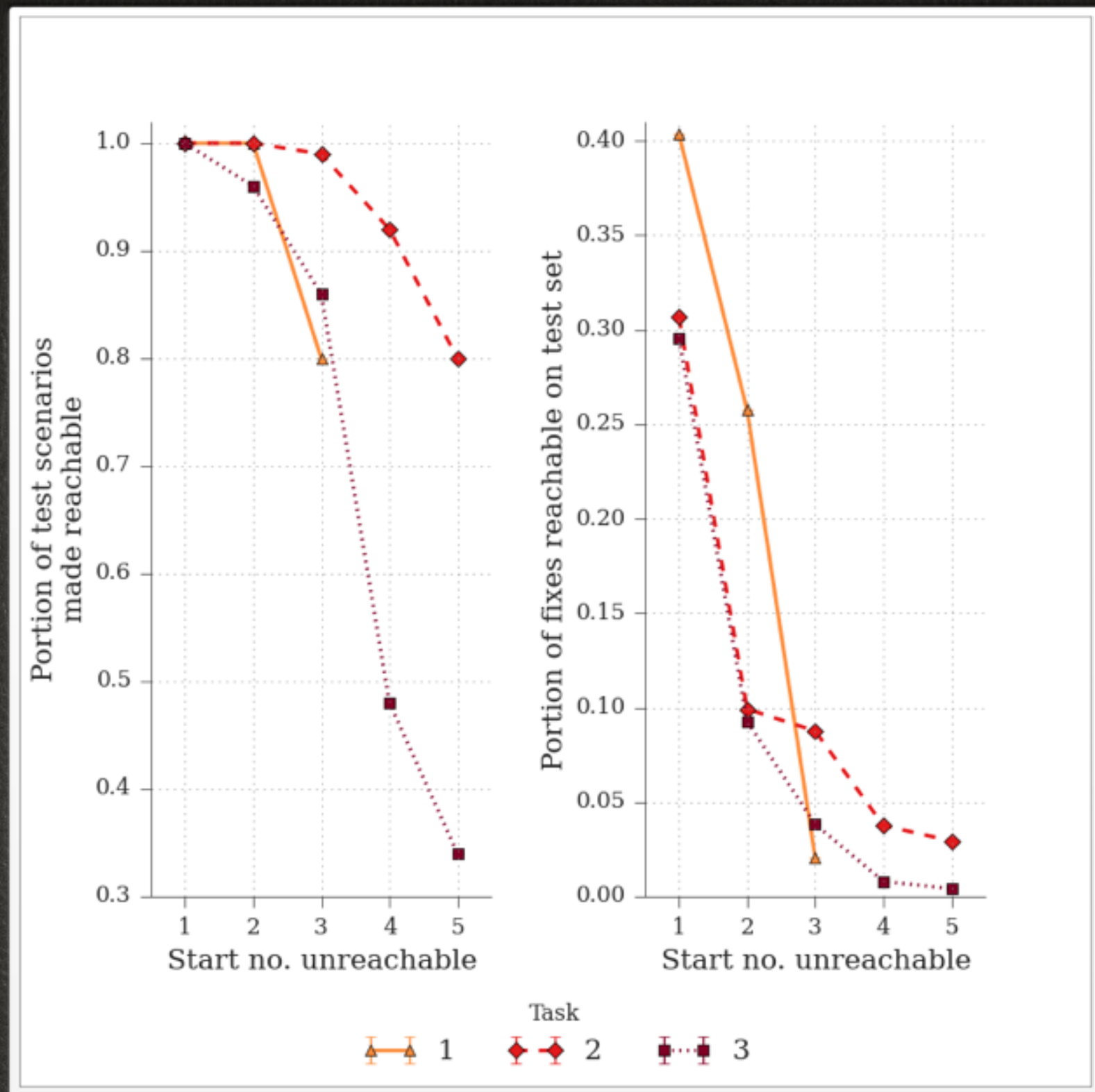
12: GREY LINES, FONTS



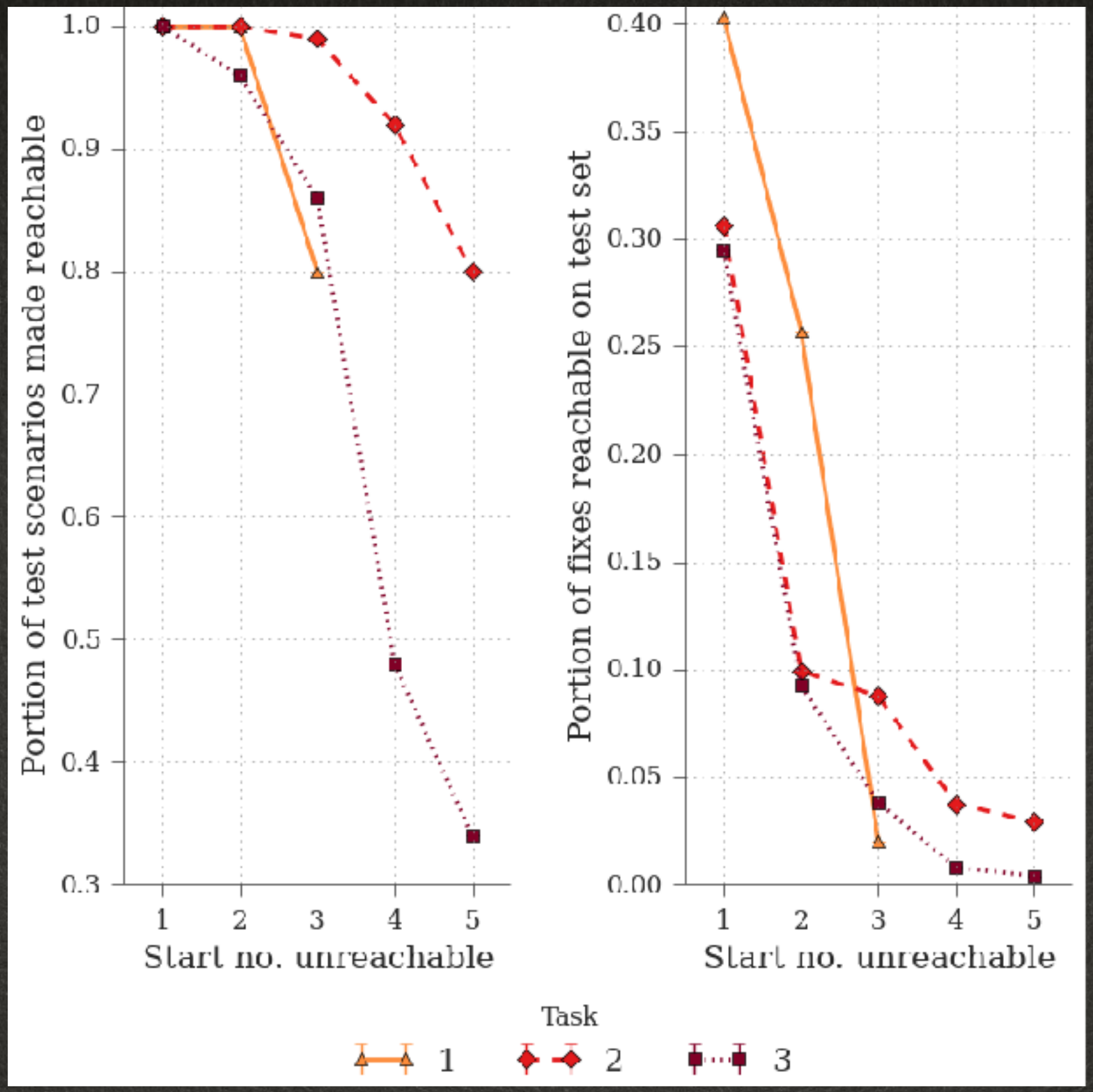
13: LINE STYLES



14: SUBPLOT SPACING

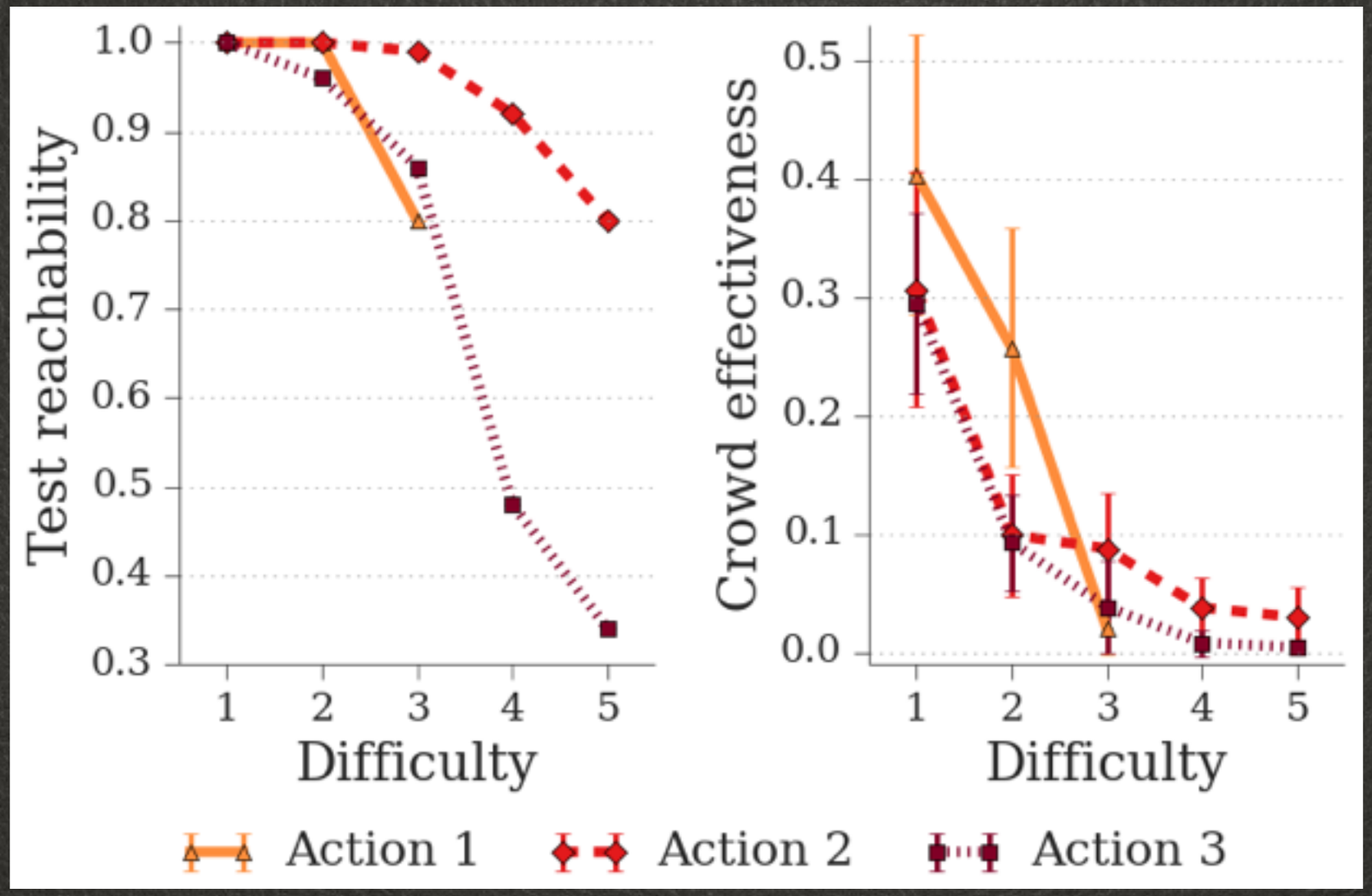


15: SPACING FOR LEGEND

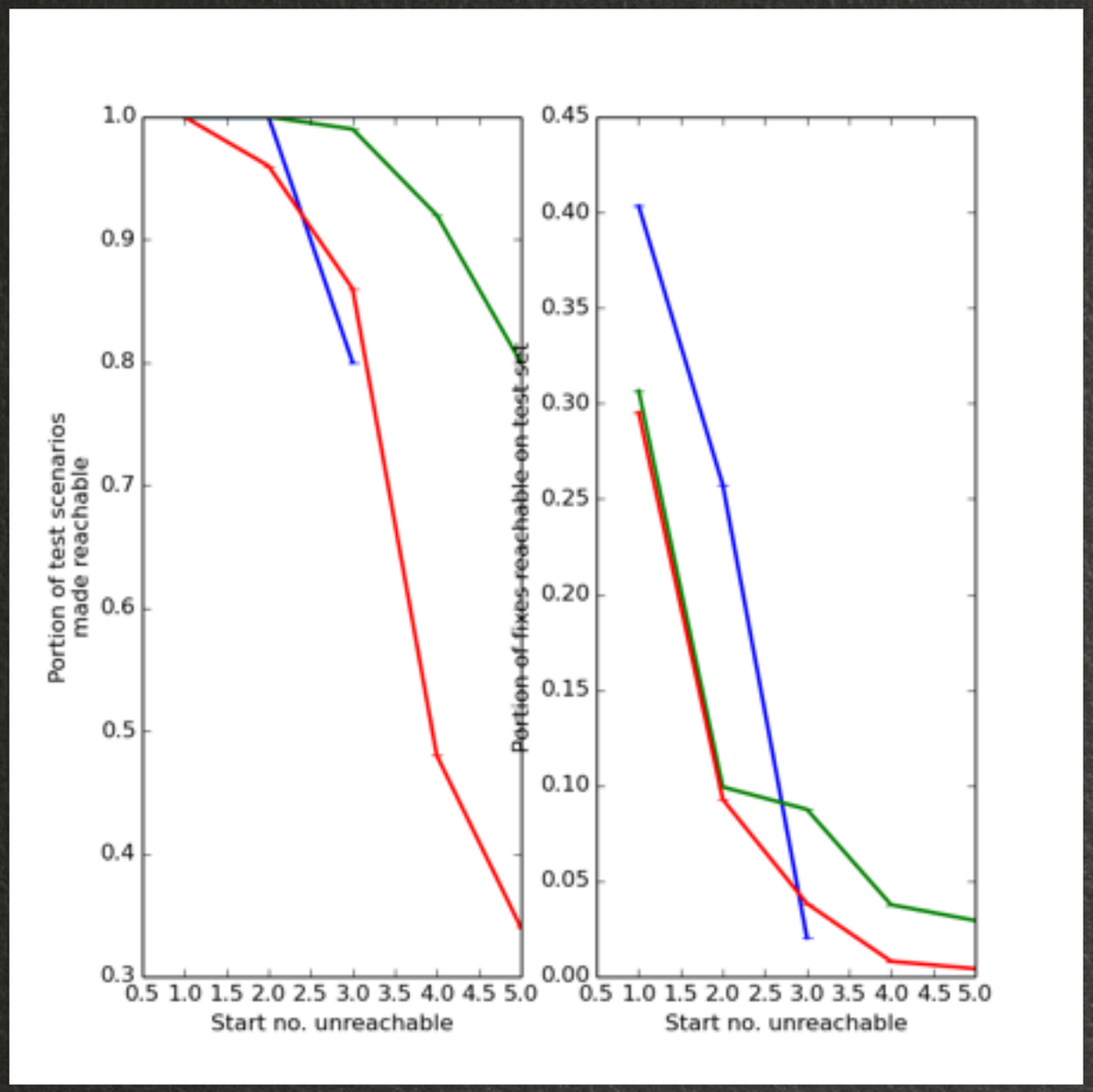


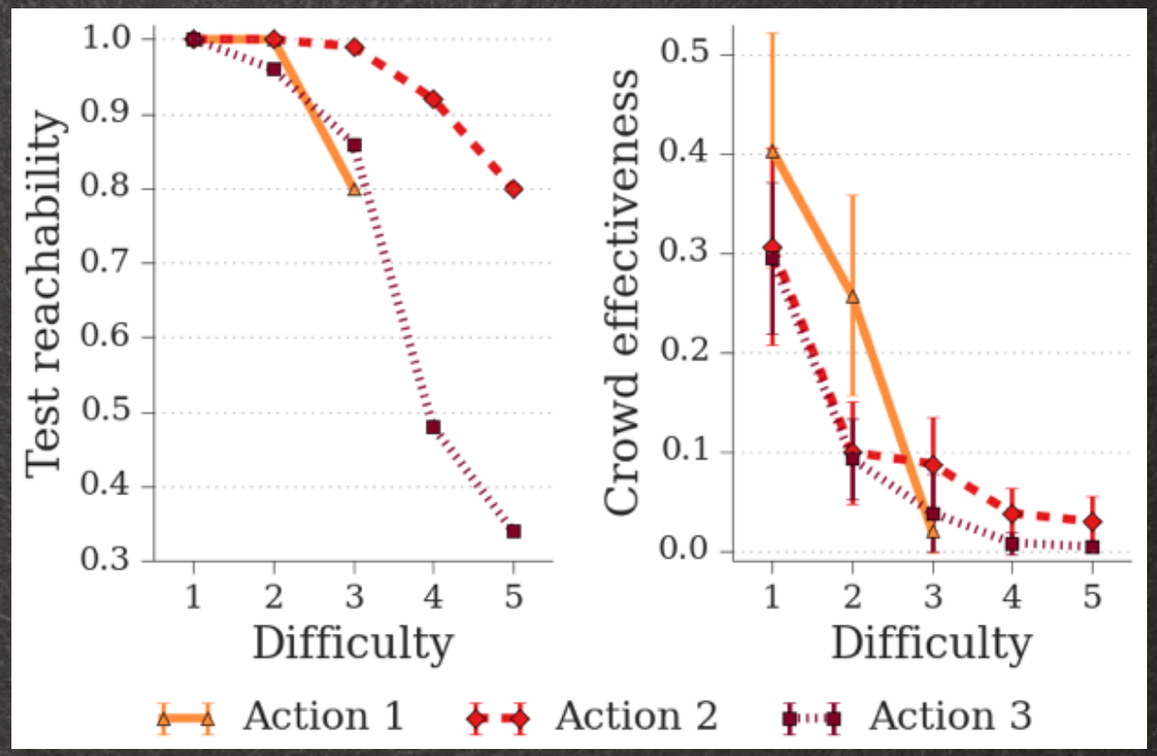
16: REMOVE BORDER

(FINAL)



17: BETTER LABELS, SHRINK


Portion of test scenarios made reachable
Start no. unreachable
Portion of fixes reachable on test set
Start no. unreachable


Test reachability
Difficulty
Crowd effectiveness
Difficulty
Action 1
Action 2
Action 3

COMPARISON

4. HOW DO I DO THIS?

I'm actually interested.

matplotlib

0. ASSUMING MATPLOTLIB

Customizing matplotlib ¶

The `matplotlibrc` file

matplotlib uses `matplotlibrc` configuration files to customize all kinds of properties, which we call `rc settings` or `rc parameters`. You can control the defaults of almost every property in matplotlib: figure size and dpi, line width, color and style, axes, axis and grid properties, text and font properties and so on. matplotlib looks for `matplotlibrc` in three locations, in the following order:

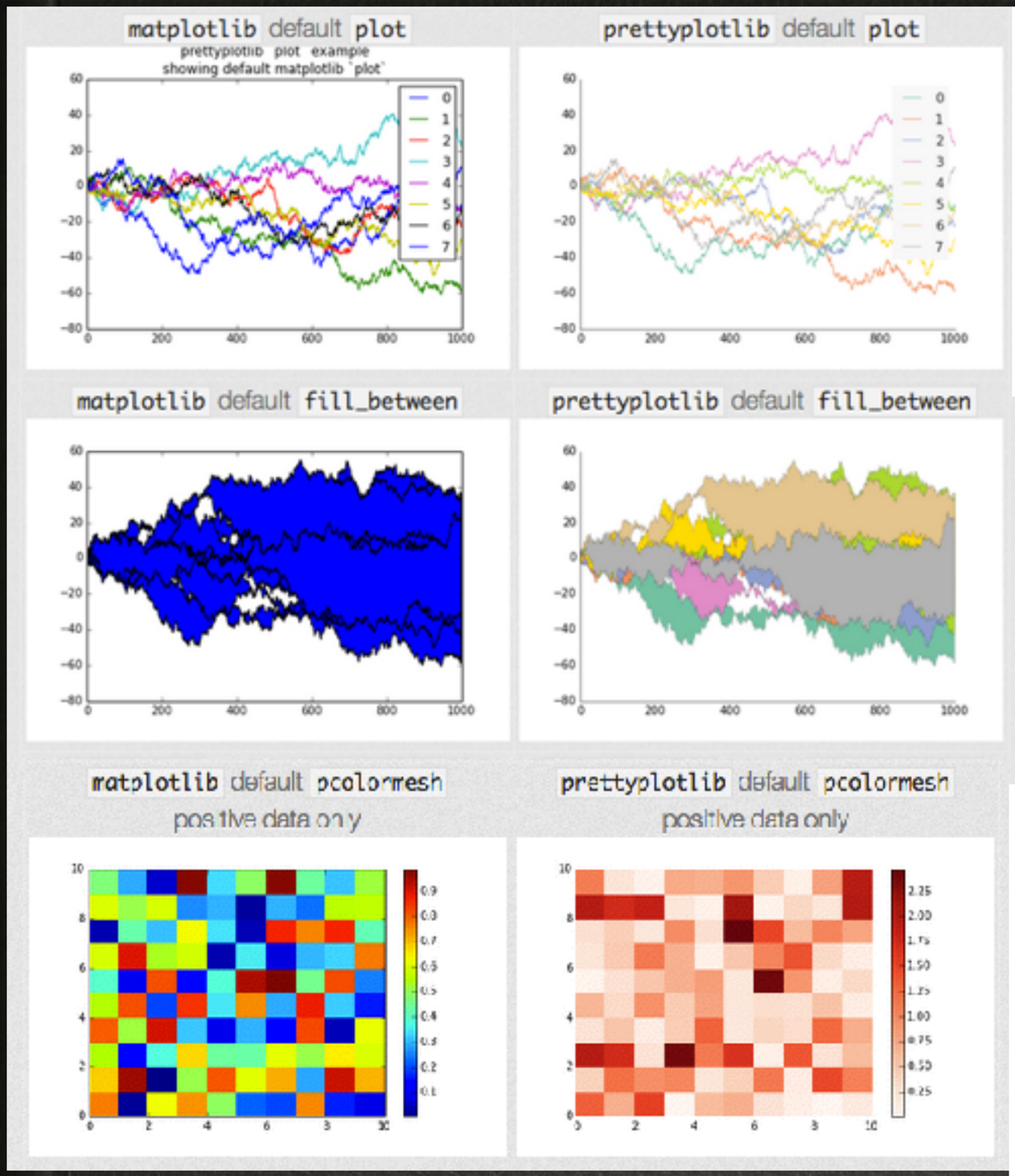
1. `matplotlibrc` in the current working directory, usually used for specific customizations that you do not want to apply elsewhere.
2. It next looks in a user-specific place, depending on your platform:
   - On Linux, it looks in `.config/matplotlib/matplotlibrc` (or `$XDG_CONFIG_HOME/matplotlib/matplotlibrc`) if you've customized your environment.
   - On other platforms, it looks in `.matplotlib/matplotlibrc`.

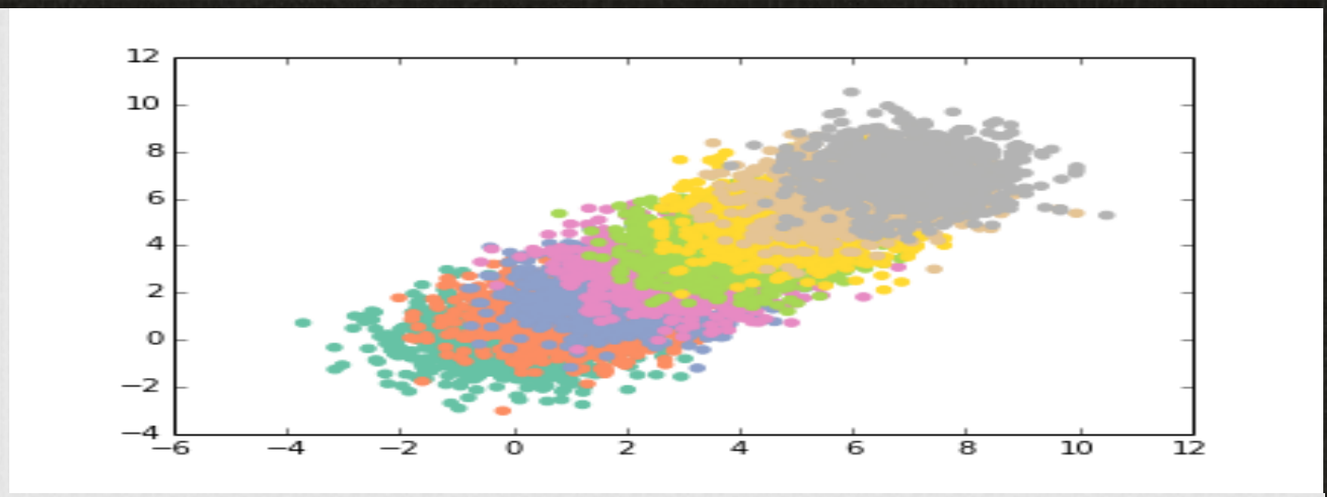
   See *.matplotlib directory location*.
3. *`INSTALL`*`/matplotlib/mpl-data/matplotlibrc`, where *`INSTALL`* is something like `/usr/lib/python2.5/site-packages` on Linux, and maybe `C:\Python25\Lib\site-packages` on Windows. Every time you install matplotlib, this file will be overwritten, so if you want your customizations to be saved, please move this file to your user-specific matplotlib directory.

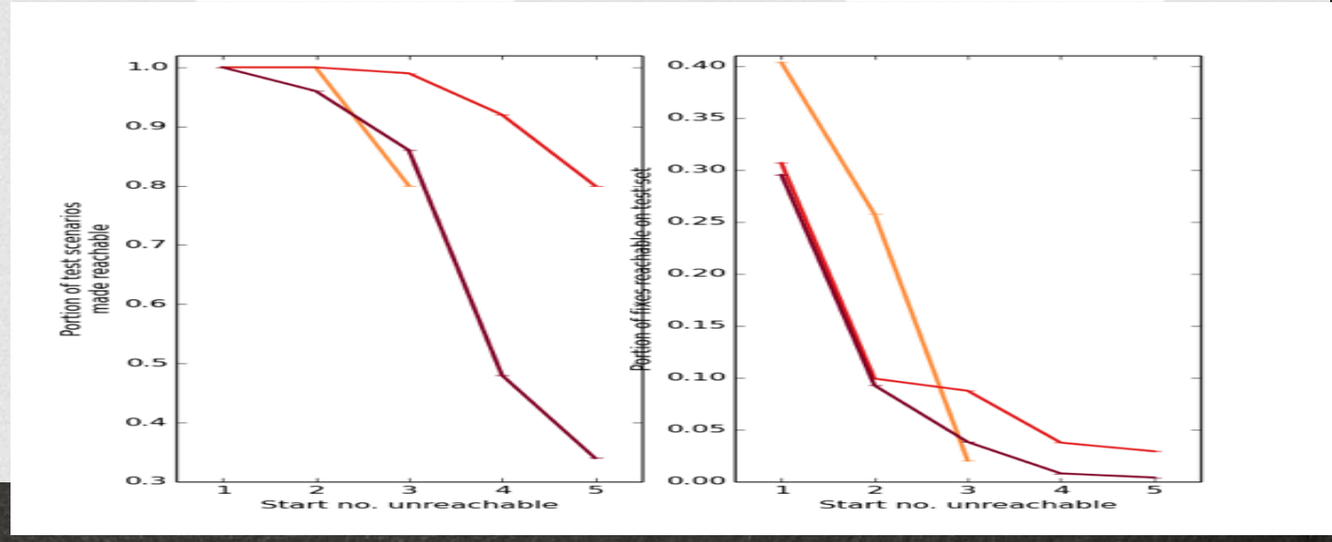
To display where the currently active `matplotlibrc` file was loaded from, one can do the following:

```
>>> import matplotlib
>>> matplotlib.matplotlib_fname()
'/home/foo/.config/matplotlib/matplotlibrc'
```

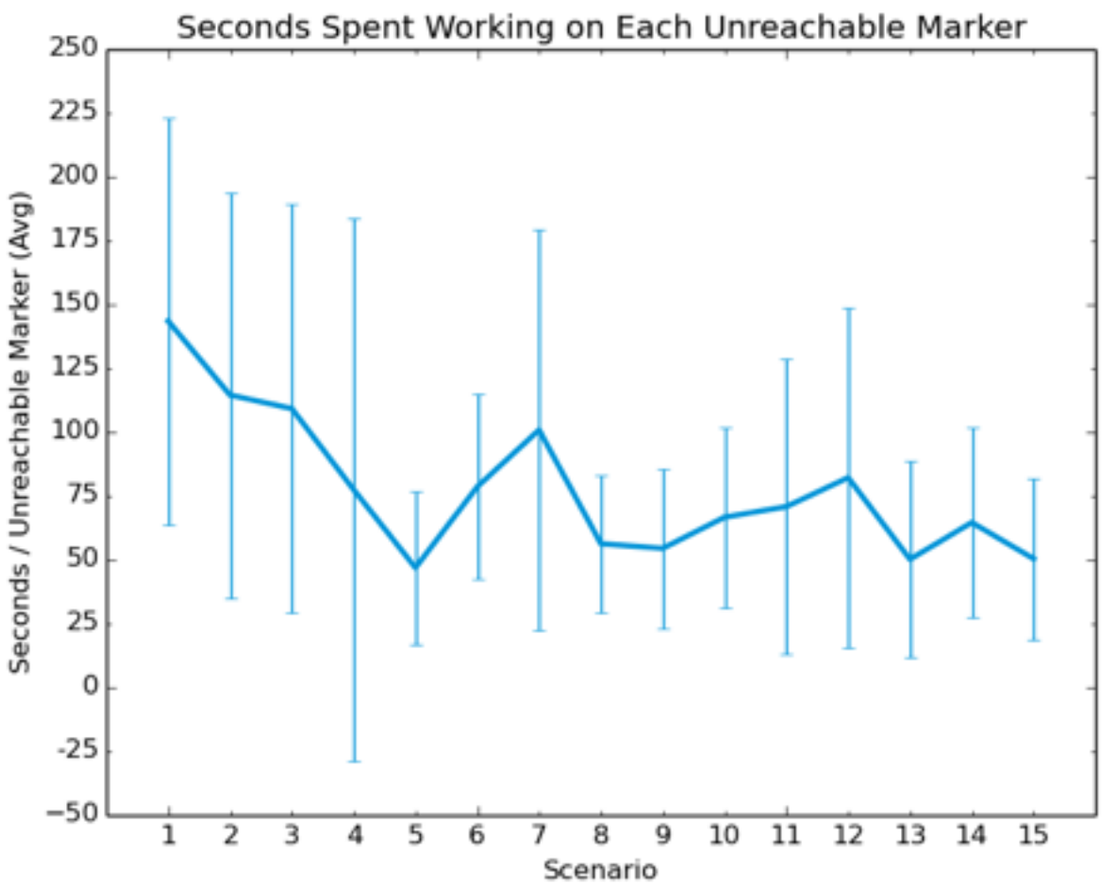
1. USE MATPLOTLIBRC

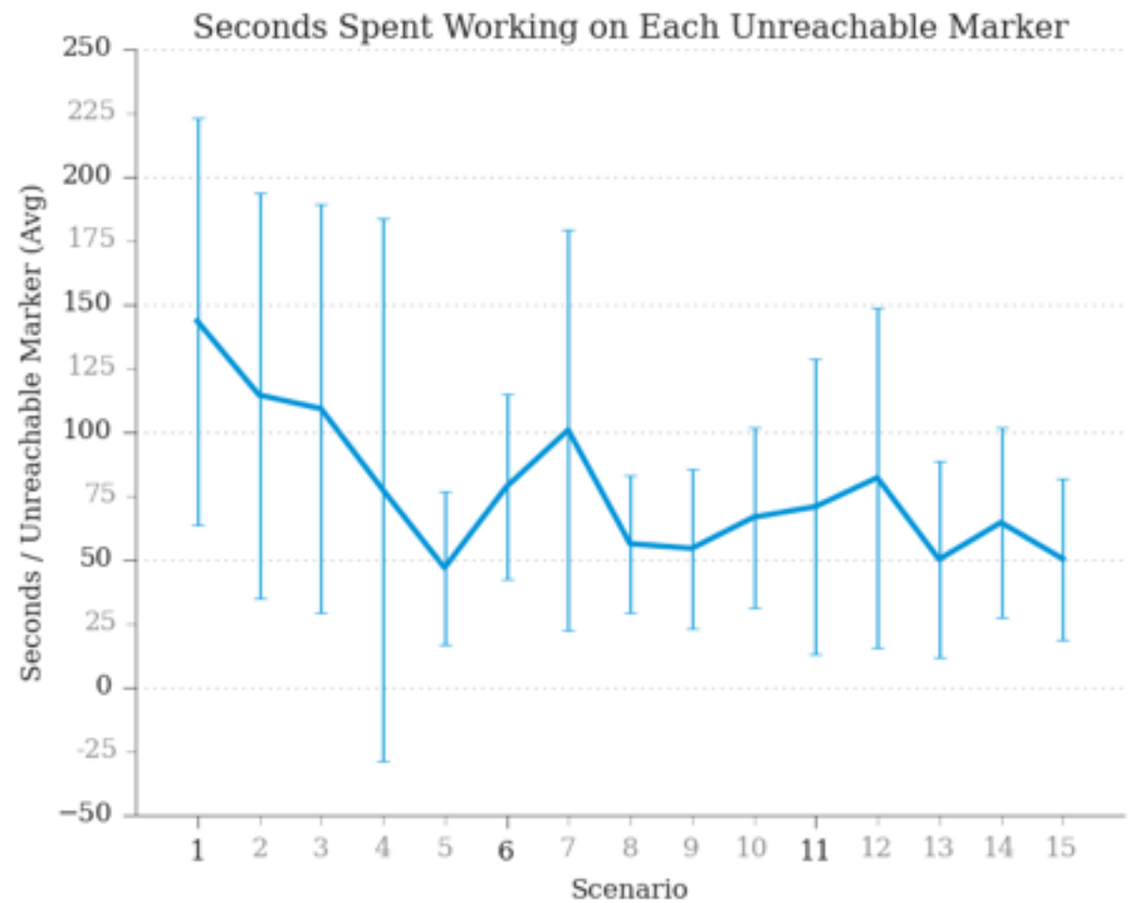






2. USE PRETTYPLOTLIB





3. USE BEAUTYPLOT

matplotlib

stackoverflow

4. USE API REF.
… & STACK OVERFLOW

Guide?

A smattering of choice functions.

Links

I made:

- Bilge Mutlu Figure Analysis (googdoc)
- beautyplot (github)

Others made:

- matplotlib (webpage) (1st tutorial)
- matplotlibrc (docs)
- prettyplotlib (github) (blog)
- making pretty graphs (blog post) [matlab, but look @ figures]
- color brewer (webpage)
- colorbrewer→matplotlib (github)