Program Comprehension Does Not Primarily Rely On the Language Centers of the Human Brain

Shashank Srikant[*] Anna A. Ivanova[†] Yotaro Sueoka[†]

Hope H. Kean[†] Riva Dhamala[‡] Evelina Fedorenko[†]

Marina U. Bers[‡] Una-May O'Reilly[*]

ABSTRACT

Our goal is to identify brain regions involved in comprehending computer programs. We use functional magnetic resonance imaging (fMRI) to investigate two candidate systems of brain regions which may support this – the Multiple Demand (MD) system, known to respond to a range of cognitively demanding tasks, and the Language system, known to primarily respond to language stimuli. We devise experiment conditions to isolate the act of *code comprehension*, and employ a state-of-the-art method to locate brain systems of interest. We administer these experiments in Python (24 participants) and ScratchJr (19 participants) - which provides a visual interface to programming, thus eliminating the effect of text in code comprehension. From this robust experiment setup, we find that the Language system is not consistently involved in code comprehension, while the MD system is. Further, we find no other brain regions beyond those in the MD system to be responsive to code. We also find that variable names, the control flow used in the program, and the types of operations performed do not affect brain responses. We discuss the implications of our findings on the software engineering and CS education communities.

KEYWORDS

Neuroimaging, fMRI, Human brain, Code comprehension, Language system, Multiple Demand system, Human factors, Python, ScratchJr

ACM Reference Format:
Shashank Srikant, Anna A. Ivanova, Yotaro Sueoka, Hope H. Kean, Riva Dhamala, Evelina Fedorenko, Marina U. Bers, and Una-May O'Reilly. 2018. Program Comprehension Does Not Primarily Rely On the Language Centers of the Human Brain. In *Neuroscience of program comprehension.* ACM, New York, NY, USA, 16 pages. https://doi.org/10.1145/1122445.1122456

[*] CSAIL, MIT [†] BCS and McGovern Institute for Brain Research, MIT [‡] Eliot-Pearson Department of Child Study and Human Development, Tufts University Correspondence to: shash@mit.edu, unamay@csail.mit.edu

Woodstock '18, June 03–05, 2020, CSAIL, MIT
© 2018 Association for Computing Machinery.
ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

Reading and understanding computer programs (code) has been estimated to consume nearly 60% of a software professional's time [58]. Yet, we understand little of how we cognitively accomplish it, making this an open question in science. Extending seminal precedents [27, 54], we attempt in this work to study and establish the regions of the brain that are involved in comprehending computer code.

The recency of code comprehension as a cognitive skill suggests that brain regions which specialize in supporting other cognitive activities likely also support code comprehension. Given its association with logic and problem solving, code comprehension can arguably be handled by regions responsible for working memory and cognitive control, or those involved in math and logic. Similarly, code and natural language share many common properties. They possess similar syntactic and semantic structures, and hierarchically compose to convey meaningful information – in both code and text, tokens are associated to form statements, which are further associated to form an entire code or document, which results in meaning being associated with the artifact [25]. Arguably, regions of the brain involved in processing language can support code comprehension.

Neuroimaging research is well positioned to address which regions are involved in code comprehension. Techniques such as functional magnetic resonance imaging (fMRI) measure brain activity when performing cognitive tasks like reading or hearing music. Brain regions whose functions have been well established, like language or music centers, responding to a new task, like code comprehension, can indicate the cognitive processes likely associated with that task [43].

In this work, we use fMRI to study how code-reading related tasks engage two known systems of brain regions – the Multiple Demand (MD) and Language systems (details in Section 3). While previous neuroimaging studies have also investigated brain regions involved in code comprehension, their results remain inconclusive. They provide evidence for activity in regions that roughly correspond to the MD system [27, 34, 42, 54, 55], as well as in regions resembling the Language system [54, 55]. Importantly, these studies do not distinguish the act of code comprehension from other code-reading related activities like mentally simulating code. Further, most do not quantify brain responses, and compare them to responses to other tasks associated with working memory or language to meaningfully interpret their observations. We review these works in Sections 2 and contrast their design choices to ours in Section 4.

Our contributions in this work are twofold. First, we design novel experiments and introduce improved methods to identify brain regions involved in code comprehension. Second, we present a new set of results which adds to our current understanding of the cognitive bases of code comprehension. We summarize our design and method contributions below. See Section 6 for details on our results.

- We offer a clearer definition of code comprehension, and design experiment conditions to isolate and measure it.
- We use a state-of-the-art procedure to determine which known, well-characterized brain systems respond to code comprehension.
- We test our experiments in two programming languages - Python and ScratchJr, a programming system with a fully visual interface, on a group of 24 and 19 participants respectively. Using ScratchJr enables measuring the effect of text on code comprehension, and additionally helps validate the generalizability of our results. Prior studies have experimented only with one programming language.
- We ensure that the observations we make generalize to different code properties like control flow (sequential programs, loops, conditionals), or types of operations performed (string, math operations).
- We additionally investigate whether brain activity corresponding to code in the Language system is a result of descriptive variable names used in codes.
- We make our code, stimuli, and brain data publicly available for the community to reuse and extend. Link - https://anonymous.4open.science/r/9fa26dd4-2011-4c3c-8739-fbd43a068244/

2 RELATED WORK

The question of whether there exist specialized regions in the human brain which are exclusive to specific cognitive functions goes back to Paul Broca's investigations of language understanding in the 1850s [31]. Advances in technology to accurately measure neural activity in the last three decades have revealed the existence of specialized regions for a variety of cognitive functions like language processing, face recognition, navigation *etc.* [39]

The use of neuroimaging techniques to study the cognitive responses to programming has gained momentum recently. Prior works have investigated the neural processes involved in debugging [14], variable tracking when reading programs [36, 46], semantic cues or program layout [18, 51], program generation [41], manipulating data structures [34], biases in code review processes [33], and programming expertise [27, 35, 48].

Relevant to our scope are works which investigate regions of the brain involved in comprehending code (as opposed to writing code, or any other coding-related activity).

Siegmund *et. al.* [54], an influential work which pointed the community's attention to this topic, investigate the question of which regions in the brain are involved in code comprehension. They present two sets of stimuli to 17 participants in an fMRI study. The first requires participants to read through snippets of code and determine their outputs. The second requires them to read code snippets with syntax errors and suggest fixes. The authors contrast activations from these two sets of stimuli, both of which correspond to code comprehension activity in the brain, to a baseline of no activity. They show parts of this contrast to lie in the Broca's region (language centers) as defined by Brodmann's areas [12].

Floyd *et. al.* [27] pose a different primary research question. They investigate, on a larger sample of 29 participants, whether it is possible to distinguish the act of program comprehension from English sentence comprehension using brain activity measurements. Their decoding experiments show that neural representations for code are unique and different from language. As a secondary result, they do comment on brain regions involved, and partially confirm Siegmund *et. al.*'s findings. In their design, they use a baseline contrast of an English comprehension task and two code-reading tasks.

Liu *et. al.* [42] very recently showed that code comprehension has very low overlap with the language centers of the brain, in line with the results we present in this work. They present 17 expert programmers with two code-related tasks - the first is similar to Siegmund *et. al.*, where participants determine code output. The second requires participants to memorize what they call 'fake code' – code snippets with scrambled tokens in each line – and confirm the presence of a specific substring. They further administer math, logic, and language tasks to locate brain regions involved in these functions in every participant. They report an overlap of code activity with regions belonging to the MD system but not the language centers.

We pose the same question that Siegmund *et. al.* and Liu *et. al.* study. We differ though in our experiment design and workflow. We compare our design choices to these works in detail in Section 4. We shall see that this leads to a different set of conclusions than those of Siegmund *et. al.*

3 BACKGROUND

We provide a brief background on fMRI studies, what is measured by such scanning machines, and the regions of the brain we investigate.

3.1 fMRI studies

Functional magnetic resonance imaging (fMRI) is typically used to identify regions of the brain which respond to any cognitive task (comprehending code, in our case). MRI machines can mark out and show brain responses in the order of a million voxels while sampling every few seconds [28]. A voxel is roughly the 3-dimensional equivalent of a pixel, and spans a few cubic millimeters of our brains.

When a brain region is involved in a cognitive task, blood flows into the region to aid its processing. An MRI machine measures this change in blood-flow, and reports BOLD (blood oxygen level dependent) values sampled at the machine's frequency. Following common practice, the parameters of a general linear model, fit to these time-varying values, are used as a metric for brain activity. We provide details in Section 5.

3.2 Regions of Interest (ROIs)

We investigate whether two well-studied systems of brain regions – the MD system and the Language system, which we know how to locate, are also activated when we comprehend code. A *region* (also referred to as *parcel*) here denotes a contiguous chunk of brain mass involved in a cognitive task. A *system* of regions (also referred to as a *network*) can comprise multiple disjoint (at the cortical level) regions, all involved in the same cognitive task.

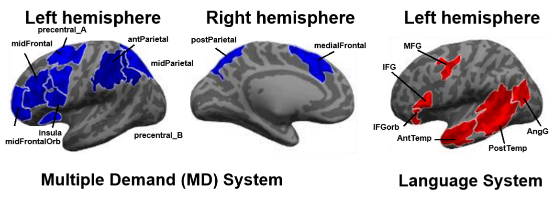


Figure 1: The Multiple demand (MD) system and Language system highlighted in a neurotypical adult brain. These two systems span multiple, closely situated regions in the brain, and have been established to have very different response profiles. What is conventionally referred to as Broca's region includes portions of both these systems [21].

Multiple Demand (MD) system. Since programming conceivably involves arithmetic and general logic skills, we investigate whether the Multiple Demand system [16], the most prominent system known to support these skills, is activated. Generally located in the prefrontal and parietal areas of the brain, this system of regions is known to be domain-agnostic, and is activated in a host of tasks requiring working memory and general problem solving skills, including math and logic [4, 16].

Language system. Another possible candidate for processing code is the Language system. These regions have been identified to respond to both comprehension and production of language across modalities (written, speech, sign language), respond to typologically diverse languages (> 50 languages, from across 10 language families), form a functionally integrated system, reliably and robustly track linguistic stimuli, and have been shown to be causally important for language [9, 10, 15, 24, 45, 52].

Figure 1 shows approximate locations of these systems in a neurotypical adult brain. These systems have been consistently located roughly in the same parts of the brain across individuals [23, 24]. While an ROI provides a set of broad regions observed to be involved in a cognitive task across individuals, we further locate *functional* ROIs (fROIs) – specific voxels within these broad regions which respond to working memory and language respectively *in an individual*. By doing this, we account for the exact anatomical locations of these voxels, which vary across individuals. This is one improved aspect of our experiment method over prior works. We provide details on fROIs in Section 4.4.

4 EXPERIMENT DESIGN

We first provide a summary of our overall workflow. We follow that with details on three key components of our experiment design: **condition design** - the various design choices we consider in creating the code stimuli we show our participants, **fMRI tasks** - the tasks participants respond to in an MRI machine which enable measuring brain activities, and **processing fMRI data** - how we analyze participants' fMRI data and quantify the effect of *code comprehension*. In our description of these components, we also contrast how they differ from previous works.

4.1 Experiment workflow - An overview

The first step of our workflow is to frame hypotheses and design conditions which can test those hypotheses. These conditions inform the stimuli and tasks we present to human participants in an MRI machine. Our goal is to observe the effect reading code has on two regions of interest in our brains - the MD system and the Language system. We first determine which voxels (fROIs) belong to the MD and the Language systems in each participant. We do this by getting participants to respond to *localizer tasks* – tasks which have been shown to consistently activate the two systems [9, 24]. We then show participants stimuli corresponding to our own carefully designed code conditions, and we measure brain responses to these conditions within the identified fROIs. The goal of analyzing fMRI responses to our code conditions is to evaluate whether they activate the fROIs as much as the localizer tasks. If they do activate the regions as much, we infer that the fROIs are involved in processing code. For example, if comprehending code activates the Language system as much as comprehending English text (the localizer task for the Language system), we then conclude that the Language system is involved in processing code comprehension in addition to processing language comprehension.

4.2 Condition design

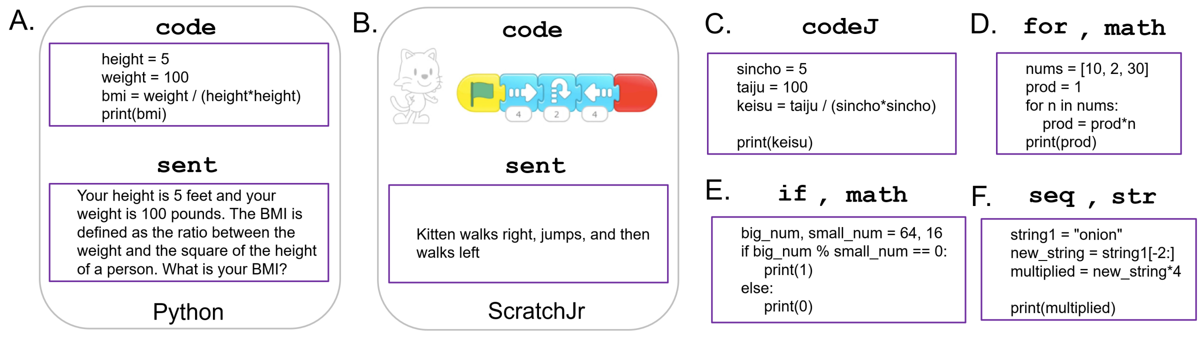
Brain activity measurements for a given condition (*e.g.* response to reading codes) can meaningfully be interpreted only relative to another condition (*e.g.* response to reading plain text), *i.e.* by *contrasting* two or more conditions. We describe the different conditions we design and contrast in our work, and discuss them in light of the design choices made by prior works.

Controlling for `non-codes`. The simplest condition pair to observe the effect of reading code is by contrasting `codes` with `non-codes` (notated as `code > non-code` in the cognitive neuroscience literature). Here, `non-codes` correspond to stimuli which participants can comprehend despite not being code-like. In our study, they correspond to statements in a natural language (English).

Our goal though is to push farther. We design conditions which help isolate the effect of other factors which might alternatively explain the activations we observe in different brain regions when understanding code.

Controlling for *code simulation*. Arguably, the task of reading code involves more than the act of *code comprehension*. To appreciate why, consider the different cognitive steps involved in reading and understanding code. On being presented code – 1) Retinal cells are activated by the presence of characters in a program 2) The visual system of our brain processes these characters. 3) Having recognized the characters, our brain interprets tokens present in the text. 4) Our brain groups tokens to recognize program statements, and eventually groups these statements to form a mental representation of the entire code, and understands its goal. 5) Our brain executes or simulates it to derive its final output.

In our work, we do not study the effects of reading code on the visual system (steps 1-2). We identify steps 3-4 as *code comprehension*, and step 5 as carrying out *code simulation*– which has also been referred to as *program tracing* [56], and *processing code content* [37]. For example, comprehending the statement `x=10+20` refers to associating this statement with the notion 'x stores the sum of



```
height = 5
weight = 100
bmi = weight / (height*height)
print(bmi)
```

Your height is 5 feet and your weight is 100 pounds. The BMI is defined as the ratio between the weight and the square of the height of a person. What is your BMI?

Kitten walks right, jumps, and then walks left

```
sincho = 5
taiju = 100
keisu = taiju / (sincho*sincho)

print(keisu)
```

```
nums = [10, 2, 30]
prod = 1
for n in nums:
    prod = prod*n
print(prod)
```

```
big_num, small_num = 64, 16
if big_num % small_num == 0:
    print(1)
else:
    print(0)
```

```
string1 = "onion"
new_string = string1[-2:]
multiplied = new_string*4

print(multiplied)
```

Figure 2: (A) A `code` condition stimulus in Python and its equivalent `sent` condition, which describes the `code` stimulus in words. `sent` controls for brain responses to *code simulation*. The difference in these conditions, `code`>`sent`, estimates *code comprehension*. (B) An example `code` and `sent` stimulus in ScratchJr, a programming system with a visual interface. ScratchJr allows to measure the effect of text in codes. (C) `codeJ` condition with Japanese variable names, which controls for the effect of meaningful variable names. (D, E, F) Conditions that measure the effect of control-flow properties (`for`, `if`, `seq`) and type of operations (`math`, `str`).

numbers 10 and 20'. *Code simulation* in this case refers to mentally adding numbers 10 and 20 and realizing that `x` stores 30. We refer to steps 3-5 collectively as *code reading*.

Step 5 can potentially dominate brain measurements made when reading and understanding code. To factor out its effect, we offer the following insight – it is possible to describe code in different ways while retaining its *code simulation* operations. A code described in sentences or as a flow diagram does not alter its operations. Drawing on this insight, we design sentences whose content matches our code conditions. We notate this condition as `sent` and the contrast as `code > sent`. See Figure 2.A. for an example. If the code condition measures *code comprehension* and *code simulation* as the dominant cognitive steps involved, the `sent` condition then arguably measures natural language (sentences) processing and *code simulation*. The difference in these two conditions `code > sent` thus allows us to isolate and measure the act of *code comprehension*.

Controlling for variable names. If *code comprehension* is indeed treated like language comprehension and the Language system is found to respond to it, it is reasonable to question whether the Language system responses are caused just by the presence of meaningful variable names and not other aspects of the code. We control for this possibility by replacing variable names with those which mean nothing in that context. The responses to such codes can then be attributed solely to *code comprehension* and not to the presence of meaningful English words in the code. In our work, we chose to rename variables with their Japanese equivalent names (written out in the English script) and administer it to participants with no knowledge of Japanese. We refer to this condition as `codeJ`. Figure 2.B shows the code in Figure 2.A instead with Japanese variable names. We also account for the effect meaningful string literals (*e.g.* `x="hello"`) or meaningful keywords (`for`, `if`) may have, by designing an equal number of stimuli without these artifacts (discussed in the following point).

Effect of control flow and operations. We additionally investigate whether brain activations to code are consistent across different code properties. This helps demonstrate the robustness of our observations to common variations possible in code. We test two such properties – control flow, and the types of operations. In control flow, we test each of loops (`for`), conditional statements (`if`), and sequential statements. See Figures 2.D, E, F for examples of each of these conditions. We test two types of operations – math and string. Figures 2.E, F show examples of `math` and str operations respectively. Every stimulus in these conditions has exactly one each of the three control structures, and one of the two data operations. This design also accounts for the presence of meaningful string literals and keywords by allowing us to observe brain activity corresponding to conditions that do not contain these artifacts (`math`, `seq` respectively).

Effect of text in codes. We experiment with the conditions we describe above in two programming languages – Python and ScratchJr. ScratchJr is a programming system with a fully visual interface [7]. It is generally introduced to children as means to express themselves creatively, where the visual interface and intuitive drag-and-drop features representing different programming constructs enable them to code without relying on a language like English [8]. The very nature of this visual interface allows us rule out the influence of text on *code comprehension*. Figure 2.B shows an example. Further, using ScratchJr as a second programming language helps validate the generalizability of our findings. All prior works have evaluated their findings only in one programming language.

Design choices by prior works. Floyd *et. al.* also use the basic contrast `code > non-code`, but nothing more to isolate *code comprehension*. Siegmund *et. al.* instead contrast `code > code with syntax errors` (Section 2). Codes with syntax errors are still codes, and hence do not help differentiate activity in regions where non-codes (natural language) are known to be processed. Further, the `code-with-syntax-errors` condition likely measures aspects of *code comprehension*, *code simulation*, and perhaps other skills specific to debugging and finding such errors. Thus, their contrast does not fully isolate *code comprehension*. While Liu *et. al.* ensure their code stimuli generalize to loops and conditions, their primary contrast `code > fake code` also does not distinguish between *code* *comprehension* and *code simulation*. Their setup introduces the additional effect of memorizing `fake code` which involves multiple cognitive processes

Summary. To summarize, in our Python experiments, our overall experiment design is a $3 \times 3 \times 2$ study – 3 conditions - `code`, `sent`, `codeJ` (Japanese variable names), and within each of these three conditions, we further have 3 categories of control flow conditions, and 2 categories of operations-related conditions. Since many of these conditions are not applicable to ScratchJr (variable names, operation types), we evaluate only the critical `code > sent` condition in ScratchJr.

4.3 fMRI tasks

For each participant, in addition to presenting stimuli corresponding to code-related conditions in an MRI machine, we present two separate tasks to *localize* the two regions of interests in them. What is central to a localizer task is its ability to strongly activate a region of interest in every individual. It has been empirically established that reading semantically well-formed sentences in any natural language strongly activates the Language system, while performing spatial memory tasks strongly and distinctly activates the MD system [23, 24]. We reuse these established localizer tasks in our work. We provide details in Section 5.

We now describe how we use this localization information when analyzing brain activity during code comprehension.

4.4 Locating fROIs and data analysis

We analyze brain data in the following five key steps. Our procedure follows the Group-constrained Subject-Specific (GSS) method of locating functional regions of interest (fROIs) that are activated consistently across individuals [47].

1. Mapping to an exemplar brain structure. To normalize differences in brain anatomies, each participant's brain is spatially transformed to an exemplar brain structure like the Montreal Neurological Institute (MNI) template [57]. These spatially transformed coordinates are used for subsequent analyses.

2. Selecting ROIs. Regions of interest (ROIs) mark out a set of broad regions observed to be involved in a cognitive task across individuals. For every participant, we use these regions as a starting point, and look for voxels within them which respond to a cognitive task. This helps avoid looking in regions which are not germane to the task. For example, reading code will understandably also activate the visual cortex, which is not of interest to our particular study.

In our work, we reuse a set of 20 MD parcels (10 in each hemisphere) and six Language parcels defined in prior works [23, 24]. These parcels have been curated by aggregating ~200 participants' brain responses to spatial working memory and language tasks respectively. As an alternate, one could select ROIs from the parcels defined by Brodmann's areas [12], an atlas which maps regions of an exemplar's brain to cognitive functions.

3. Identifying fROIs. For every individual, a *functional* region of interest (fROI) refers to a collection of voxels within an ROI which respond to the cognitive task the ROI is involved in. Owing to differences in anatomies, the specific set of voxels which respond to a cognitive task (like spatial reasoning or language) varies across individuals. ROIs, aggregated from across individuals, help narrow down the search space to locate these specific voxels in every individual by pointing to a swath of regions known to respond to the task. fROIs in turn identify specific voxels *functionally involved* in the cognitive task. Localizer tasks (Section 4.3) for each system help identify these voxels. By the end of this step, we establish in each participant fROIs for the MD and the Language systems. We provide details in Appendix 1.

4. Aggregate activation data within a participant. We use the fROIs defined for the two systems in the previous step in all our remaining experiment conditions. Specifically, we measure the activations of our code conditions in the selected fROIs. At this stage, we have at least two sets of activation measurements for each voxel in an fROI – one corresponding to the localizer task, and the others corresponding to the different code-related conditions. For each fROI, we obtain a single response value per condition by averaging the responses of all voxels within the fROI.

5. Aggregate activation data across participants. For each system, we then evaluate whether the distribution of participant-level responses to the code conditions is comparable to that of the localizer task. If it is, we conclude that the system is involved in processing code conditions.

Multi-participant analysis without functional localizers. Among prior works, Liu *et. al.* alone use localizer tasks to find task-selective voxels in individual participants. However, they do not use ROIs (step 2 above) and instead perform a whole-brain analysis, and report overlaps as against measuring exact activations in fROIs. Their setup coupled with their ambiguous condition design (discussed in Section 4.2) makes it hard to infer brain regions accurately.

In fMRI studies which do not use localizer information, as in the case of Siegmund *et. al.*, Floyd *et. al.*, and other works which have studied different aspects of programming, the primary difference is that ROIs are defined based on anatomy, and not on their function (*i.e.* how they respond to localizer tasks). Concretely, this difference arises in steps 3 and 4, where instead of aggregating activations within an fROI, activations are estimated in each voxel across the entire brain and aggregated across participants (also called the *group analysis* procedure). The location of such aggregated active voxels is then described using anatomical labels, such as Brodmann areas [12]. This method has broadly been referred to as *reverse inference* in neuroimaging studies [49].

The reverse inference method assumes that fROIs are spatially fixed among individuals and can be uniquely located in the exemplar brain structure. While reverse inference is not always a concern, especially when the regions are anatomically well separated and distinct (*e.g.* visual system vs. MD system), it has been shown to yield inaccurate estimates in the measurements of the closely situated MD and the Language systems [5, 11, 21, 22]. What is referred to as the language region by Brodmann's areas (areas 44 and 45) in one individual can instead refer to functional regions belonging to the MD system in another individual, owing to differences in individual anatomies [20, 22]. The GSS approach of function-based ROI identification helps circumvent this potential cause for inaccuracy.

5 EXPERIMENT PROCEDURE

We describe in brief our experiment procedure. We provide details in Appendix A.

We recruited 24 participants for Experiment 1 (Python) and 19 participants for Experiment 2 (ScratchJr), with no overlap between these groups. On the day of the scan, having provided consent, participants spent 1.5 – 2 hours in the scanner. In Experiment 1, in the week of their scheduled fMRI scan, each participant additionally completed an assessment in Python to evaluate their fluency in it (Appendix A.1).

Once in the scanner, a participant was presented with two localizer tasks, adopted from prior works [23, 24], to locate the MD system and Language system respectively in their brain (Appendix A.2). The MD system localizer task is a working memory task, presented in two grades of difficulty - easy and hard. The Language system task has two conditions - sentence reading (SR), and non-word reading (NR). SR requires reading sentences which are structurally and semantically meaningful. NR requires reading sentences with pronounceable yet meaningless non-words (*e.g.* `BIZBY ACWORILLY BUSHU SNOOKI BILIBOP KUKEE`). These two conditions serve as references to measure other experiment conditions against – the Language system has been shown to respond strongly to SR while only minimally to NR.

Participants were also presented with coding tasks (Appendix A.3). The tasks shown were balanced between the three conditions - `codeE` (code with semantically meaningful variable names in English), `sent` (sentences describing programs, controlled for *code simulation*), and `codeJ` (code with Japanese variable names). Each participant saw 72 problems, 24 from each of the three conditions. Each of these set of 24 problems further had an equal number of control-flow and operations-related conditions. Any given participant saw only one of the three versions of a problem (Appendix A.4).

The data from the localizer scans was used to locate the fROIs in the MD and the Language systems in every participant (Appendix A.5). We fit a general linear model to the time series brain activation data generated as a response to our different tasks. The parameters of this model (β) are used as a metric for brain activity (BOLD) in all our analyses (Appendix A.6).

6 RESULTS

We present our questions and their corresponding results here. In our results, we discuss the neural activations in different regions of the brain (Figures 3.A, 3.B). The x-axis in these plots corresponds to the different conditions participants responded to, and the y-axis represents activation strength (β values, see Appendix A.6 for details). Each dot in each bar corresponds to one participant's aggregate activity in the fROIs localized in them. When reporting results of a contrast between any two conditions, we measure the difference in the average β values ($\Delta\beta$) and compute its associated p-value.

RQ 1. Does *code reading* activate the Multiple Demand (MD) system?
Conditions contrasted. `code`, `sentence reading`, `non-word reading`

We begin by investigating whether reading code, which involves both *code comprehension* and *code simulation*, activates the MD system. We do this by comparing the activations of our primary code-related condition - code problems, to the localizer tasks for the Language system - sentence reading and non-word reading. We notate these conditions as CP, SR, and NR respectively in Figure 3.A, B. We evaluate two sets of fROIs in the MD system - one in each hemisphere of the brain (Figure 3.A, B., left and center plots). From the plots, we see both sentence reading and non-word reading, the language localizer conditions, have minimal activations in the MD system in both hemispheres. This is expected since the MD system is not sensitive to language tasks [9]. We find that code problems, which account for both *code comprehension* and *code simulation*, activate fROIs in both hemispheres of the MD system consistently and significantly more than the baselines in both our experiments (Python: $\Delta\beta$= 2.17, $p < 0.001$; ScratchJr: $\Delta\beta$= 1.23, $p < 0.001$). This suggests that the MD system is involved in reading code.

We confirm whether these responses are consistent across code properties, which will establish its robustness to the variations possible in code. We test two properties – control-flow (sequential, `for`, `if`), and types of data manipulated in them (string, math operations) in Python. We observe strong responses regardless of the type of operations and control flow (Figure 3.C; y-axis is response to CP). We thus conclude that the responses in the MD system to code problems were not a result of any one particular type of problem, or mental operations related to a particular control flow.

This clearly identifies and establishes the role of the MD system in *code reading*. Prior works did not identify and study this system of regions.

RQ 1 result. Yes, *code reading* activates the MD system. Its responses are independent of the control-flow operations and types of data operations present in codes.

RQ 2. Does *code comprehension* activate the Multiple Demand (MD) system?
Conditions contrasted. `code`, `sent`, `sentence reading`, `non-word reading`

Since we find that *code reading* activates the MD system, we investigate whether these were responses to *code comprehension* or *code simulation*. To answer this, we study the effect of both our code-related conditions – code problems (CP), and sentence problems which match the code problems for their content (SP). We find that sentence problems, which measure only *code simulation* and not *code comprehension*, activate the MD system significantly greater than the language localizer baselines in both hemispheres only for Python (left: $\Delta\beta$= 1.51, $p < 0.001$; right: $\Delta\beta$= 0.78, $p < 0.001$). This activation is not significant for ScratchJr (left: $\Delta\beta$= 0.09, p = 0.93; right: $\Delta\beta$= -0.40, p = 0.004), suggesting that *code simulation*

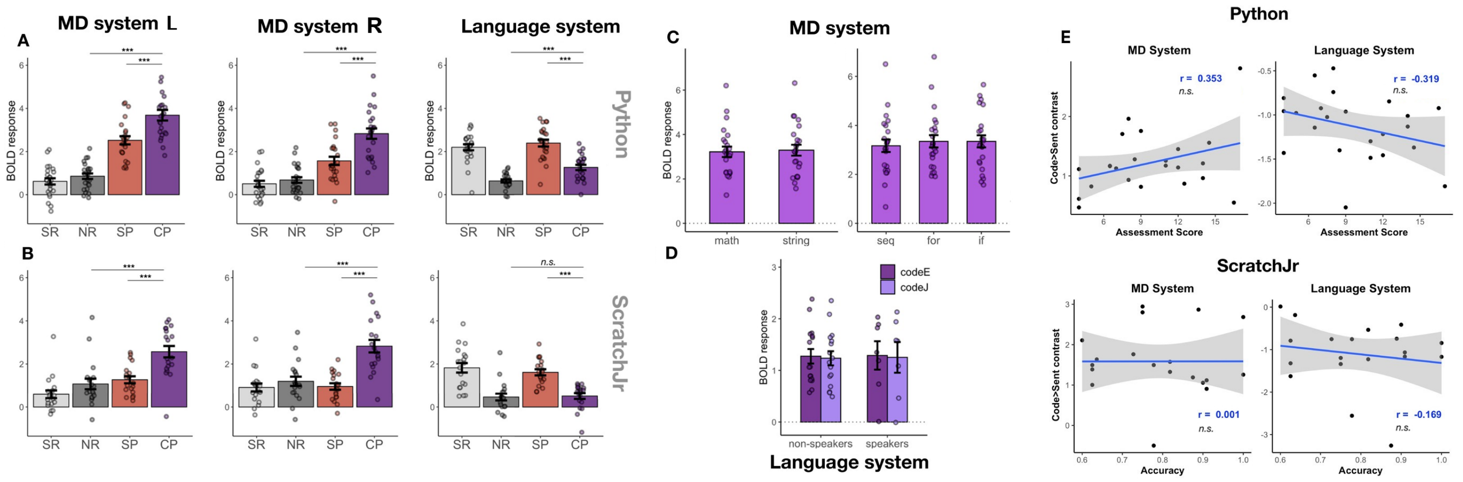


Figure 3: (A, B) Brain activations in the MD system left hemisphere (MD system L), MD system right hemisphere (MD system R), and the Language system. We measure responses to four conditions – codes (CP), sentences matching the code's operations (SP), Sentence reading (SR), and Non-words reading (NR). We experiment in Python (N=24) and ScratchJr (N=19). Each dot in the bars corresponds to aggregate data from one participant. * indicates $p < 0.001$, `n.s.` - not significant (C) MD system responses to two code properties – operation type (math, string operations), and control-flow (sequential, loop (`for`), conditional (`if`)) (D) Language system responses to variable names in English (`codeE`) and Japanese (`codeJ`) (E) Correlation of responses in the MD and the Language systems to proficiency in Python (top) and ScratchJr (bottom).**

is not consistently supported by the MD. However, we find that code problems, which measure both *code comprehension* and *code simulation*, strongly activate fROIs in both hemispheres. This is despite sentence problems taking slightly longer on average to respond to (Appendix B.2). We hence find that CP strongly activates the MD system and SP does not. This implies that the difference CP > SP, which measures *code comprehension*, strongly activates it. This is strong evidence for the MD system's consistent and robust activation to *code comprehension*, and shows it is not just a response to the underlying *code simulation* operations.

We investigate further for any hemispheric bias towards *code comprehension*. Previous works have shown that math and logic problems typically activate the MD system in the left-hemisphere of the brain [3, 4]. We did not find any such bias in Python (MD-L plot, Figure 3.A). In ScratchJr, we observe stronger responses in the right hemisphere ($\Delta\beta$= 0.57, p < 0.001; MD-R plot, 3.B), perhaps reflecting a known bias of the right-hemisphere towards visuo-spatial processing [53].

Follow up analyses of activity within individual regions within the MD system showed that 17 of the 20 fROIs in the Python experiment, and 14 of the 20 fROIs in the ScratchJr experiment responded significantly more strongly to code problems than to sentence problems (details in Appendix B.1). This demonstrates code processing is broadly distributed across the MD system and is not localized to a particular subset of regions within it. Within this activated subset, we evaluate whether any fROIs are *selective* to code problems in comparison to other cognitively demanding tasks which activate the MD system. We find none for ScratchJr, and three regions in the frontal lobe (precentral-A, precentral-B, midFrontal) which exhibit stronger responses to Python code problems than to the hard working memory localizer task for the MD system. However, the magnitude of `code > sent` in these regions ($\Delta\beta$= 1.03, 0.95, 0.97) was comparable to the mean magnitude across all MD system fROIs (average $\Delta\beta$= 1.03), suggesting that the high response was caused by the underlying *code simulation* rather than *code comprehension*. We conclude that *code comprehension* is broadly supported by the MD system, and no specific regions in the MD system are functionally specialized for it.

These new results further establish the role of the MD system in processing *code comprehension*, which we narrowly and clearly define in this work.

RQ 2 result. Yes, *code comprehension* consistently activates the MD system. Unlike math and logic, it activates fROIs in both the left and right hemispheres. In fact, no specific fROI within the MD system specializes for *code comprehension*, and it is instead broadly supported by the entire system.

RQ 3. Does *code reading* activate the Language system?
Conditions contrasted. `code`, `sent`, `sentence reading`, `non-word reading`

We investigate the Language system similarly for responses to our code conditions. Figures 3.A, B (rightmost plot) show the aggregate responses in the Language system to the two code conditions and the two language localizer conditions described above. As expected of the localizers, we find the activations of sentence reading to be significantly greater than non-word reading [9, 24]. Among the code conditions, we find that sentence problems activate the Language system as much as the sentence localizer task in both Python and ScratchJr. This is again expected since sentence problems contain English sentences describing what the program does (Figure 2.A). However, the responses to code problems were weaker than responses to sentence problems in both experiments (Python: $\Delta\beta$= 0.98, $p < 0.001$, ScratchJr: $\Delta\beta$= 0.99, $p < 0.001$). This observation alone does not yield any insight on whether code activates the Language system, and we hence compare these activations to the localizer baseline non-word reading. Non-word reading is a lower bound for activity in the Language system; this is the activity seen in the Language system when it is not actively engaged in linguistic interpretation. Responses to the code condition were stronger than non-word reading only in the Python experiment ($\Delta\beta$= 0.78, $p < 0.001$) but not in the ScratchJr experiment ($\Delta\beta$= 0.15, p= 0.29), implying that code does not consistently activate the Language system.

The result from this principled investigation of the Language system is contrary to that of Siegmund *et. al.*, who report the involvement of the language system in addition to other brain regions. We discuss this further in Section 7.

RQ 3 result. No, *code reading* does not consistently activate the Language system.

RQ 4. Do meaningful variable names affect the Language system's response to code?
Conditions contrasted. `codeE`, `codeJ`

Since we find that Python code activates the Language system but ScratchJr does not, we investigate whether this is a consequence of meaningful variable names present in codes. To study this effect, we had participants read half the Python code problems with semantically meaningful variable names in English (`codeE`) and the other half with Japanese variable names (`codeJ`), making them semantically meaningless; 18 of the 24 participants reported no knowledge of Japanese. In the Language system, we found no effect of meaningful variable names ($\Delta\beta$= 0.03, p = 0.84) (Figure 3.D, non-speakers), knowledge of Japanese ($\Delta\beta$= 0.03, p = 0.93) (Figure 3.D, speakers), nor any interaction between the two ($\Delta\beta$= 0.09, p = 0.71), suggesting that the Language system response was not affected by the presence of semantically meaningful variable names. This result is surprising since the Language system has been shown to be very sensitive to word meaning [6]. A possible explanation is that participants do not fully engage with the words' meanings to solve problems.

RQ 4 result. Meaningful variable names do not affect the Language system's response to code.

RQ 5. Are there regions outside the MD system and Language system that respond to *code comprehension*?

To search for regions responsive to *code comprehension* outside the MD system and Language system, we perform a whole-brain Group-constrained Subject Specific analysis. For both Python and ScratchJr, we search for brain areas with activations where `code` > `sent`. We then examine the response of such regions to code and sentence problems (cross-validated with held-out data), as well as to conditions from the two localizer experiments. In both experiments, the discovered regions spatially resembled the MD system. For Python, any region that responded to code also responded to the spatial working memory task (MD system localizer). In case of ScratchJr, some fROIs responded more strongly to code problems than to the spatial working memory task; these fROIs were located in early visual areas/ventral visual stream which likely responded to low-level visual properties of ScratchJr code (which contains colorful icons, objects, *etc.*). These whole-brain analyses demonstrate that the MD system responds robustly and consistently to *code comprehension*, confirming the results of the fROI-based analyses discussed in RQs 1 and 3. This further shows that fROI-based analyses did not miss any non-visual regions outside the boundaries of the MD and the Language systems that was activated by *code comprehension.*

RQ 5 result. We found no code-selective regions outside the MD and the Language systems.

RQ 6. Does expertise affect how the MD and the Language systems respond to *code comprehension*?

We study the role of expertise by correlating responses within each system with independently obtained proficiency scores for participants of Experiment 1 (a 1-hour Python assessment module; Appendix A.1) and with in-scanner accuracy scores for Experiment 2 participants. Figure 3.E plots the percentage proficiency scores (x-axis) against *code comprehension* (`code` > `sent`). No correlations were significant. However, due to a relatively low number of participants (N = 24 and N = 19, respectively), these results should be interpreted with caution.

RQ 6 result. We did not have enough data to observe the effect of expertise on the MD and the Language systems' responses to *code comprehension.*

7 DISCUSSION

We present a new set of results which improves our understanding of the cognitive bases of program comprehension. We find that *code reading* (which we identify to include both *code comprehension* and *code simulation*) strongly activates only the MD system and not the Language system. Despite their anatomical proximity in the left-hemisphere of our brains, our work clearly distinguishes the roles of both these systems by means of functional localizers.

MD system results. We find that the MD system consistently processes *code reading*. We support our observations by showing these activations generalize across two code properties - control flow and data operations, suggesting that the system's response is robust to variations in code. We further learn that the MD system responds consistently to *code comprehension.* It also responds to *code simulation* strongly in Python, but we see only a weak evidence for it in ScratchJr, which needs to be investigated in future work. It is reasonable to expect the MD system to process *code reading*, since both *code comprehension* and *simulation* requires attention, working memory, inhibitory control, planning, and general flexible relational reasoning - cognitive processes long linked to the MD system [16, 17]. This finding also supports Liu *et. al.*'s recent results [42]. Since no other regions outside the MD system responded to codes, we posit this system stores code-specific knowledge in addition to processing it. This knowledge likely includes concepts specific to a programming language (*e.g.* the syntax marking an array in Java vs. Python) and concepts shared across languages (loops, conditions). Evidence from studies on processing mathematics and physics [4, 26] has shown that the MD system can store some domain-specific representations in the long term, perhaps for evolutionarily late-emerging and late-acquired domains of knowledge. In conclusion, we identify a known brain system, which had previously not been studied for its role in *code reading* tasks, to be involved in *code comprehension* specifically.

Language system results. We importantly establish in this work that *code reading* is *not* consistently processed by the Language system. This is a new finding, and adds to the results from Siegmund *et. al.* and Floyd *et. al.*, while confirming results from Liu *et. al.*. Siegmund *et. al.* report the involvement of the language centers in *code comprehension* by showing evidence of left-lateralized brain activity. While it is unclear whether their observations were technically from the Language system or the MD system, we suspect that they observed *code simulation* and not *code comprehension.* Additionally, and surprisingly, we find that the Language system is insensitive to the presence of meaningful variable names. More work is required to determine why the Language system showed some activity in response to Python code.

The Language system does not respond consistently to *code comprehension* despite numerous similarities between code and natural language. However, the lack of consistent Language system engagement in *code comprehension* does not mean that the mechanisms underlying language and code processing are completely different. It is possible that both the MD and the Language systems have similarly organized neural circuits that allow them to map a symbol to a concept. However, the fact that we observed code-related activity primarily in the MD system indicates that *code comprehension* does not activate the same neural circuits as language, and needs to use domain-general MD system circuits instead.

Having identified the regions activated when reading programs, we discuss how our results affect the programming languages (PL), software engineering (SE), and CS education (CS-Ed) communities.

Impact on the PL, SE, CS-Ed communities. To understand how our results can be applied specifically to *improving* how we can understand programs, we first establish the relationship between two cognitive activities engaging the same brain system (in our case - working memory tasks and *code comprehension* engaging the MD system). A few studies have claimed that for any two cognitive activities that share the same brain resources, training one activity will lead to an improvement in the other [38, 44]. For example, if language and music share and activate the same brain system, then tools and approaches used to engage and train one activity should be transferable to, and will lead to an improvement in the other. Since the effects of training and improving one's MD system are not well understood, it is unclear whether training on cognitively demanding non-coding tasks could improve our ability to read and understand programs.

Based on the opposite conclusions presented in Siegmund *et. al.*, Portnoff *et. al.* [50] and similar other works suggest adopting a "languages first" approach when teaching programming. Evidence from our work does not support this claim, and we caution against adopting practices that are used to teach natural languages for programming just based on conclusions from recent neuroimaging studies.

The Language system not being involved in *code comprehension* should not diminish the role of language in understanding programs. The use of poorly named variables has been shown to increase cognitive load [19], and non-native English speakers have been found to often struggle with learning English-based programming languages [30]. Future work should reconcile this disparity, and aim to show how results from studies on cognition can aid understanding programs.

ML models for code. Recent advances in machine learning (ML) models trained on large corpora of programs have shown models' ability to perform tasks like renaming functions, bug detection, *etc.* [2]. Deep networks learn a 'language model' of programs, and likely learn a generalized way to represent these programs. Do these model-learned representations correspond to the representations (activation data) in the different fROIs from our study? Such a correspondence may have far reaching implications. For instance, if we can probe and isolate specific weights or layers that encode loops and recursion in a recurrent model like seq2seq, code2seq, or GPT-3, the existence of a correspondence between representations may help locate the encodings of loops and recursion in our brains. Such a correspondence has recently been established between representations stored in the visual cortex and those learned by deep convolutional networks for image processing and recognition [13, 40, 59]. This promises to be a compelling direction for future work.

8 THREATS TO VALIDITY

There are limitations to the results we report in this work. One possible threat is posed by the tasks we designed – do our programming tasks measure code comprehension and understanding? The programs we present in this study are short snippets of procedural code with limited program properties. They do not have complex control and data dependencies generally seen in production-grade programming projects. Properties like function calls, objects, types, complex data, and state changes in the program are not included either and should be studied in the future, building on the understanding of simpler snippets provided by our work. Further, we study a very specific activity related to programming – reading programs, and do not investigate other equally important aspects like designing solutions, selecting appropriate data structures, and writing programs.

In designing our code stimuli, having overtly informative variables names poses the risk of participants not going through all the lines of code presented to them, and instead just guessing what the code does by gleaning variable names. To avoid this, we constrain our variable names to be informative and natural (as it would appear in a real codebase) to the extent they do not reveal fully the intent of the entire code snippet. However, such a constraint does not appear in actual coding scenarios. Disparate stimuli are a source of possible confounds. If some conditions had disproportionately longer code than others, it would be unclear if any trend we saw in brain activations were a function of the condition, or such factors like code length. In an attempt at avoiding this, we ensured that the stimuli in our $3 \times 3 \times 2$ conditions had similar code lengths and overall response times. However, they are not all equal.

Expertise is also a potential confound which could affect the generalizability of the results we see. The majority of our participants were recruited from a university setting and had roughly 3-6 years of programming experience. While our participants' experience level was largely homogeneous in our study, expertise could interact with brain functions associated with program comprehension, as it does with other cognitive functions [1, 29, 32]. Accounting for the role of expertise would require running these experiments on a population with varying proficiencies.

Neuroimaging experiments generally risk lack of generalizability of results owing to low sample sizes. In our work, the relatively small sample sizes (N=24 for Experiment 1, and N=19 for Experiment 2) affect only our group analyses (of comparing aggregate information across participants). Although these sample sizes are the norm in the neuroimaging community, the robustness of our results are limited by the small number of participants. We see this as an opportunity for authors from similar neuroimaging studies to collaborate to analyze data across these works, which will also help amortize the high costs of carrying out this type of experiments.

9 CONCLUSION

Our work presents a new set of methods, experiments, and results which furthers our understanding of how our brains comprehend computer code. It is unique in addressing two core issues that made it difficult to interpret results from prior studies. First, we disentangle responses to the act of *code comprehension* from *code simulation*. Second, we analyze responses in brain areas that are functionally localized in individual participants, which provides an accurate interpretation of the observed responses. We find that the MD system in our brains consistently processes *code comprehension*, while the Language system does not. We release our code, stimuli, and brain data for the community to reuse and extend.

REFERENCES

[1] Aakash Agrawal, KVS Hari, and SP Arun. 2018. How does reading expertise influence letter representations in the brain? An fMRI study. *Journal of Vision* 18, 10 (2018), 1161–1161.

[2] Miltiadis Allamanis, Earl T Barr, Premkumar Devanbu, and Charles Sutton. 2018. A survey of machine learning for big code and naturalness. *ACM Computing Surveys (CSUR)* 51, 4 (2018), 1–37.

[3] Marie Amalric and Stanislas Dehaene. 2016. Origins of the brain networks for advanced mathematics in expert mathematicians. *Proceedings of the National Academy of Sciences* 113, 18 (2016), 4909–4917.

[4] Marie Amalric and Stanislas Dehaene. 2019. A distinct cortical network for mathematical knowledge in the human brain. *NeuroImage* 189 (2019), 19–31.

[5] Katrin Amunts and Karl Zilles. 2012. Architecture and organizational principles of Broca's region. *Trends in cognitive sciences* 16, 8 (2012), 418–426.

[6] Andrew James Anderson, Edmund C Lalor, Feng Lin, Jeffrey R Binder, Leonardo Fernandino, Colin J Humphries, Lisa L Conant, Rajeev DS Raizada, Scott Grimm, and Xixi Wang. 2019. Multiple regions of a cortical network commonly encode the meaning of words in multiple grammatical positions of read sentences. *Cerebral cortex* 29, 6 (2019), 2396–2411.

[7] Marina Umaschi Bers. 2018. Coding, playgrounds and literacy in early childhood education: The development of KIBO robotics and ScratchJr. In *2018 IEEE global engineering education conference (EDUCON)*. IEEE, 2094–2102.

[8] Marina U Bers, Carina González-González, and Mª Belén Armas-Torres. 2019. Coding as a playground: Promoting positive learning experiences in childhood classrooms. *Computers & Education* 138 (2019), 130–145.

[9] Idan Blank, Nancy Kanwisher, and Evelina Fedorenko. 2014. A functional dissociation between language and multiple-demand systems revealed in patterns of BOLD signal fluctuations. *Journal of neurophysiology* 112, 5 (2014), 1105–1118.

[10] Idan A Blank and Evelina Fedorenko. 2017. Domain-general brain regions do not track linguistic input as closely as language-selective regions. *Journal of Neuroscience* 37, 41 (2017), 9999–10011.

[11] Matthew Brett, Ingrid S Johnsrude, and Adrian M Owen. 2002. The problem of functional localization in the human brain. *Nature reviews neuroscience* 3, 3 (2002), 243–249.

[12] Korbinian Brodmann. 1909. *Vergleichende Lokalisationslehre der Grosshirnrinde in ihren Prinzipien dargestellt auf Grund des Zellenbaues*. Barth.

[13] Santiago A Cadena, George H Denfield, Edgar Y Walker, Leon A Gatys, Andreas S Tolias, Matthias Bethge, and Alexander S Ecker. 2019. Deep convolutional models improve predictions of macaque V1 responses to natural images. *PLoS computational biology* 15, 4 (2019), e1006897.

[14] Joao Castelhano, Isabel C Duarte, Carlos Ferreira, Joao Duraes, Henrique Madeira, and Miguel Castelo-Branco. 2019. The role of the insula in intuitive expert bug detection in computer code: an fMRI study. *Brain imaging and behavior* 13, 3 (2019), 623–637.

[15] David Glenn Clark and Jeffrey L Cummings. 2003. Aphasia. In *Neurological Disorders*. Elsevier, 265–275.

[16] John Duncan. 2010. The multiple-demand (MD) system of the primate brain: mental programs for intelligent behaviour. *Trends in cognitive sciences* 14, 4 (2010), 172–179.

[17] John Duncan and Adrian M Owen. 2000. Common regions of the human frontal lobe recruited by diverse cognitive demands. *Trends in neurosciences* 23, 10 (2000), 475–483.

[18] Sarah Fakhoury, Yuzhan Ma, Venera Arnaoudova, and Olusola Adesope. 2018. The effect of poor source code lexicon and readability on developers' cognitive load. In *2018 IEEE/ACM 26th International Conference on Program Comprehension (ICPC)*. IEEE, 286–28610.

[19] Sarah Fakhoury, Devjeet Roy, Yuzhan Ma, Venera Arnaoudova, and Olusola Adesope. 2019. Measuring the impact of lexical and structural inconsistencies on developersâĂŹ cognitive load during bug localization. *Empirical Software Engineering* (2019), 1–39.

[20] Evelina Fedorenko, Michael K Behr, and Nancy Kanwisher. 2011. Functional specificity for high-level linguistic processing in the human brain. *Proceedings of the National Academy of Sciences* 108, 39 (2011), 16428–16433.

[21] Evelina Fedorenko and Idan A Blank. 2020. BrocaâĂŹs area is not a natural kind. *Trends in cognitive sciences* 24, 4 (2020), 270–284.

[22] Evelina Fedorenko, John Duncan, and Nancy Kanwisher. 2012. Language-selective and domain-general regions lie side by side within BrocaâĂŹs area. *Current Biology* 22, 21 (2012), 2059–2062.

[23] Evelina Fedorenko, John Duncan, and Nancy Kanwisher. 2013. Broad domain generality in focal regions of frontal and parietal cortex. *Proceedings of the National Academy of Sciences* 110, 41 (2013), 16616–16621.

[24] Evelina Fedorenko, Po-Jang Hsieh, Alfonso Nieto-Castañón, Susan Whitfield-Gabrieli, and Nancy Kanwisher. 2010. New method for fMRI investigations of language: defining ROIs functionally in individual subjects. *Journal of neurophysiology* 104, 2 (2010), 1177–1194.

[25] Evelina Fedorenko, Anna Ivanova, Riva Dhamala, and Marina Umaschi Bers. 2019. The language of programming: a cognitive perspective. *Trends in cognitive sciences* 23, 7 (2019), 525–528.

[26] Jason Fischer, John G Mikhael, Joshua B Tenenbaum, and Nancy Kanwisher. 2016. Functional neuroanatomy of intuitive physical inference. *Proceedings of the national academy of sciences* 113, 34 (2016), E5072–E5081.

[27] Benjamin Floyd, Tyler Santander, and Westley Weimer. 2017. Decoding the representation of code in the brain: An fMRI study of code review and expertise. In *2017 IEEE/ACM 39th International Conference on Software Engineering (ICSE)*. IEEE, 175–186.

[28] Gary H Glover. 2011. Overview of functional magnetic resonance imaging. *Neurosurgery Clinics* 22, 2 (2011), 133–139.

[29] Jesse Gomez, Michael Barnett, and Kalanit Grill-Spector. 2019. Extensive childhood experience with Pokémon suggests eccentricity drives organization of visual cortex. *Nature human behaviour* 3, 6 (2019), 611–624.

[30] Philip J Guo. 2018. Non-native english speakers learning computer programming: Barriers, desires, and design opportunities. In *Proceedings of the 2018 CHI conference on human factors in computing systems*. 1–14.

[31] Victor W Henderson. 1986. Paul Broca's less heralded contributions to aphasia research: Historical perspective and contemporary relevance. *Archives of Neurology* 43, 6 (1986), 609–612.

[32] Klaus Hoenig, Cornelia Müller, Bärbel Herrnberger, Eun-Jin Sim, Manfred Spitzer, Günter Ehret, and Markus Kiefer. 2011. Neuroplasticity of semantic representations for musical instruments in professional musicians. *NeuroImage* 56, 3 (2011), 1714–1725.

[33] Yu Huang, Kevin Leach, Zohreh Sharafi, Nicholas McKay, Tyler Santander, and Westley Weimer. 2020. Biases and differences in code review using medical imaging and eye-tracking: genders, humans, and machines. In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 456–468.

[34] Yu Huang, Xinyu Liu, Ryan Krueger, Tyler Santander, Xiaosu Hu, Kevin Leach, and Westley Weimer. 2019. Distilling neural representations of data structure manipulation using fMRI and fNIRS. In *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*. IEEE, 396–407.

[35] Yoshiharu Ikutani, Takatomi Kubo, Satoshi Nishida, Hideaki Hata, Kenichi Matsumoto, Kazushi Ikeda, and Shinji Nishimoto. 2020. Expert programmers have fine-tuned cortical representations of source code. *Eneuro* (2020).

[36] Yoshiharu Ikutani and Hidetake Uwano. 2014. Brain activity measurement during program comprehension with NIRS. In *15th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD)*. IEEE, 1–6.

[37] Anna A Ivanova, Shashank Srikant, Yotaro Sueoka, Hope H Kean, Riva Dhamala, Una-May O'Reilly, Marina U Bers, and Evelina Fedorenko. 2020. Comprehension of computer code relies primarily on domain-general executive brain regions. *Elife* 9 (2020), e58906.

[38] Susanne M Jaeggi, Martin Buschkuehl, John Jonides, and Walter J Perrig. 2008. Improving fluid intelligence with training on working memory. *Proceedings of the National Academy of Sciences* 105, 19 (2008), 6829–6833.

[39] Nancy Kanwisher. 2010. Functional specificity in the human brain: a window into the functional architecture of the mind. *Proceedings of the National Academy of Sciences* 107, 25 (2010), 11163–11170.

[40] Seyed-Mahdi Khaligh-Razavi and Nikolaus Kriegeskorte. 2014. Deep supervised, but not unsupervised, models may explain IT cortical representation. *PLoS computational biology* 10, 11 (2014), e1003915.

[41] Ryan Krueger, Yu Huang, Xinyu Liu, Tyler Santander, Westley Weimer, and Kevin Leach. 2020. Neurological divide: an fMRI study of prose and code writing. In *2020 IEEE/ACM 42nd International Conference on Software Engineering (ICSE)*. IEEE, 678–690.

[42] Yun-Fei Liu, Judy Kim, Colin Wilson, and Marina Bedny. 2020. Computer code comprehension shares neural resources with formal logical inference in the fronto-parietal network. *Elife* 9 (2020), e59340.

[43] Mara Mather, John T Cacioppo, and Nancy Kanwisher. 2013. How fMRI can inform cognitive theories. *Perspectives on Psychological Science* 8, 1 (2013), 108–113.

[44] Monica Melby-Lervåg and Charles Hulme. 2013. Is working memory training effective? A meta-analytic review. *Developmental psychology* 49, 2 (2013), 270.

[45] Zachary Mineroff, Idan Asher Blank, Kyle Mahowald, and Evelina Fedorenko. 2018. A robust dissociation among the language, multiple demand, and default mode networks: evidence from inter-region correlations in effect size. *Neuropsychologia* 119 (2018), 501–511.

[46] Takao Nakagawa, Yasutaka Kamei, Hidetake Uwano, Akito Monden, Kenichi Matsumoto, and Daniel M German. 2014. Quantifying programmers' mental workload during program comprehension based on cerebral blood flow measurement: A controlled experiment. In *Companion proceedings of the 36th international conference on software engineering*. 448–451.

[47] Alfonso Nieto-Castañón and Evelina Fedorenko. 2012. Subject-specific functional localizers increase sensitivity and functional resolution of multi-subject analyses. *Neuroimage* 63, 3 (2012), 1646–1669.

[48] Chris Parnin, Janet Siegmund, and Norman Peitek. 2017. On the Nature of Programmer Expertise.. In *Ppig*. 16.

[49] Russell A Poldrack. 2011. Inferring mental states from neuroimaging data: from reverse inference to large-scale decoding. *Neuron* 72, 5 (2011), 692–697.

[50] Scott R Portnoff. 2018. The introductory computer programming course is first and foremost a language course. *ACM Inroads* 9, 2 (2018), 34–52.

[51] Ivonne Schröter, Jacob Krüger, Janet Siegmund, and Thomas Leich. 2017. Comprehending studies on program comprehension. In *2017 IEEE/ACM 25th International Conference on Program Comprehension (ICPC)*. IEEE, 308–311.

[52] Cory Shain, Idan Blank, Marten van Schijndel, Evelina Fedorenko, and William Schuler. 2019. fMRI reveals language-specific predictive coding during naturalistic sentence comprehension. *Neuropsychologia* (2019).

[53] Summer L Sheremata, Katherine C Bettencourt, and David C Somers. 2010. Hemispheric asymmetry in visuotopic posterior parietal cortex emerges with visual short-term memory load. *Journal of Neuroscience* 30, 38 (2010), 12581–12588.

[54] Janet Siegmund, Christian Kästner, Sven Apel, Chris Parnin, Anja Bethmann, Thomas Leich, Gunter Saake, and André Brechmann. 2014. Understanding understanding source code with functional magnetic resonance imaging. In *Proceedings of the 36th International Conference on Software Engineering*. 378–389.

[55] Janet Siegmund, Norman Peitek, Chris Parnin, Sven Apel, Johannes Hofmeister, Christian Kästner, Andrew Begel, Anja Bethmann, and André Brechmann. 2017. Measuring neural efficiency of program comprehension. In *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering*. 140–150.

[56] Elliot Soloway. 1986. Learning to program= learning to construct mechanisms and explanations. *Commun. ACM* 29, 9 (1986), 850–858.

[57] Nathalie Tzourio-Mazoyer, Brigitte Landeau, Dimitri Papathanassiou, Fabrice Crivello, Olivier Etard, Nicolas Delcroix, Bernard Mazoyer, and Marc Joliot. 2002. Automated anatomical labeling of activations in SPM using a macroscopic anatomical parcellation of the MNI MRI single-subject brain. *Neuroimage* 15, 1 (2002), 273–289.

[58] Xin Xia, Lingfeng Bao, David Lo, Zhenchang Xing, Ahmed E Hassan, and Shanping Li. 2017. Measuring program comprehension: A large-scale field study with professionals. *IEEE Transactions on Software Engineering* 44, 10 (2017), 951–976.

[59] Daniel LK Yamins, Ha Hong, Charles F Cadieu, Ethan A Solomon, Darren Seibert, and James J DiCarlo. 2014. Performance-optimized hierarchical models predict neural responses in higher visual cortex. *Proceedings of the national academy of sciences* 111, 23 (2014), 8619–8624.

A METHODS

In this section, we present details on the stimuli we used for the fROI localizer tasks, how we defined fROIs, stimuli we designed for the coding tasks, the procedure we followed to present these tasks to our study participants, how we recruited our participants, and finally, our analysis of the recorded MRI data.

A.1 Participants

For Experiment 1, we recruited 25 participants (13 women, mean age = 22 years, SD = 3.0). Average age at which participants started to program was 16 years (SD = 2.6); average number of years spent programming was 6.3 (SD = 3.8). In addition to Python, 20 people also reported some knowledge of Java, 18 people reported knowledge of C/C++, 4 of functional languages, and 20 of numerical languages like Matlab and R. Twenty-three participants were right-handed, one was ambidextrous, and one was left-handed; the left-handed participant had a right-lateralized Language system and was excluded from the analyses, leaving 24 participants total (the rest had left-lateralized language regions). All participants in Experiment 1 were also provided a 1-hour paper-pencil Python assessment, administered in the week of their scheduled scan. It consisted of 23 questions, 22 of which expected short-responses. One question required an algorithm to be designed and implemented in Python. There was no overlap between questions on this assessment and the stimuli presented in the experiments. We used this to measure their general proficiency in Python and programming.

For Experiment 2, we recruited 21 participants (13 women, mean age = 22 years, SD = 2.8). There was no overlap in participants between the two experiments. In addition to ScratchJr, 8 people also reported some knowledge of Python, 6 people reported knowledge of Java, 9 people reported knowledge of C/C++, 1 of functional languages, and 14 of numerical languages like Matlab and R (one participant did not complete the programming questionnaire). Twenty were right-handed and one was ambidextrous; all participants had right-lateralized language regions, as evaluated with the language localizer task. Two participants from Experiment 2 had to be excluded due to excessive motion levels during the MRI scan, leaving 19 participants total.

All participants were recruited from local universities and their surrounding communities, and compensated for their participation. All were native speakers of English, had normal or corrected to normal vision, and reported working knowledge of Python or ScratchJr, respectively. The protocol for these studies was approved by an institutional review board. All participants gave written informed consent in accordance with protocol requirements.

A.2 Localizer tasks

All participants completed a language localizer task aimed at identifying language-responsive brain regions [24], a spatial working memory localizer task aimed at identifying the multiple demand (MD) brain regions [23], and a set of coding tasks.

A language localizer task identified brain regions within individual participants that selectively respond to language stimuli. During the task, participants read sentences (*e.g.* NOBODY COULD HAVE PREDICTED THE EARTHQUAKE IN THIS PART OF THE COUNTRY) and lists of disconnected, pronounceable non-words (*e.g.* U BIZBY ACWORRILY MIDARALBUSHU SNOOKI BILIBOP KUKEE WEPS WIBRON PUZ). Each stimulus consisted a total of twelve words/non-words. For details of how the language materials were constructed, see Fedorenko *et. al.* [24]. We refer to the sentence reading task as SR and non-word reading task as NR. The SR > NR contrast has been previously shown to reliably activate left-lateralized fronto-temporal language processing regions and to be robust to changes in the materials, task, and modality of presentation [24, 45]. Stimuli were presented in the center of the screen, one word/non-word at a time, at the rate of 450 ms per word/non-word. Each stimulus was preceded by a 100 ms blank screen and followed by a 400 ms screen showing a picture of a finger pressing a button, and a blank screen for another 100 ms, for a total trial duration of 6 s. Participants were asked to press a button whenever they saw the picture of a finger pressing a button. This task was included to help participants stay alert and awake.

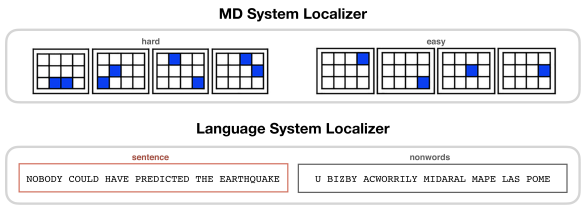

MD System Localizer
hard
easy
Language System Localizer
sentence
NOBODY COULD HAVE PREDICTED THE EARTHQUAKE
nonwords
U BIZBY ACWORRILY MIDARAL MAPE LAS POME

Figure 4: The two localizer tasks we adopted from prior works [23, 24]. The MD system localizer task tests spatial reasoning. Participants are shown, in quick succession, 4 screens with different highlighted squares in a grid. They are then shown two grids and have to correctly identify the grid which superimposes all the highlighted squares shown to them. The Language system localizer task requires participants to read two sets of sentences - one coherent and meaningful (left), and the other pronounceable yet meaningless (right).

Previous work established a spatial working memory task which identified the MD system in individuals [23]. Participants had to keep track of four (the easy condition) or eight (the hard condition) sequentially presented locations in a 3×4 grid (see Figure 2). In both conditions, they performed a two-alternative forced-choice task at the end of each trial to indicate the set of locations they just saw. This `hard > easy` contrast has been previously shown to robustly activate MD regions [23]. These regions have been shown to respond to difficulty manipulations across many diverse tasks [16, 23]. Stimuli were presented in the center of the screen across four steps. Each of these steps lasted for 1000 ms and presented one location on the grid in the easy condition, and two locations in the hard condition. Each stimulus was followed by a choice-selection step, which showed two grids side by side. One grid contained the locations shown on the previous four steps, while the other contained an incorrect set of locations. Participants were asked to press one of two buttons to choose the grid that showed the correct locations.

A.3 Coding tasks

The coding tasks in Experiment 1 (Python) included three conditions: problems in Python with English variables (`codeE`), problems in Python with Japanese variables (`codeJ`), and problems described in English sentences (`sent`). The full list of problems is available on the project's code repository. Each participant saw 72 problems, 24 from each of the three conditions. Any given participant saw only one version of a problem. Half of the problems in each condition required performing mathematical operations, and the other half required string manipulations. In addition, both math and string-manipulation problems varied in program structure: $\frac{1}{3}$ of the problems of each type included only sequential statements, $\frac{1}{3}$ included a `for` loop, and $\frac{1}{3}$ included an `if/else` statement. Participants were instructed to read the problem statement and press a button when they were ready to respond (the minimum reading time was restricted to 5 s and the maximum to 50 s; mean reading time was 19 s). Once they pressed the button, they were presented with four response options and had to indicate their response by pressing a corresponding button. The response screen was presented for 5 s.

Experiment 2 (ScratchJr) included two conditions: short programs in ScratchJr, and verbal descriptions of problems presented as written sentences. During each trial, participants were presented with a fixation cross 4 s, followed by a program for 8 s (in either code or verbal form), followed by a video that either matched or did not match that output of the program. Participants were instructed to indicate whether the video matched the description by pressing one of the two buttons as the video was playing. Average video length was 4.13 s (SD 1.70 s).

A.4 Experiment procedure

Each task was organized as follows - blocks contained multiple stimuli (referred to as *trials* in the neuroimaging literature) from a given condition. Blocks from different conditions were arranged to form a *run*. The Language system and MD system localizer tasks had two blocks - a fixation block and the experiment blocks containing the localizer conditions. Experiment blocks lasted 18 s (with 3 trials per block), and fixation blocks lasted 14 s. Each run (consisting of 5 fixation blocks and 16 experimental blocks) lasted 358 s. Each participant completed 2 runs.

For the MD localizer tasks, experimental blocks lasted 32 s (with 4 trials per block), and fixation blocks lasted 16 s. Each run (consisting of 4 fixation blocks and 12 experimental blocks) lasted 448 s. Each participant completed 2 runs.

The condition order was randomly counterbalanced across runs.

In each run of Experiment 1 (Python), participants completed 2 trials from each of the three conditions (`codeE`, `codeJ`, `sent`). Each run included 3 fixation blocks, each of which lasted 10 s. One run lasted an average of 176 s (SD = 34 s). Each participant completed 12 runs.

For Experiment 2 (ScratchJr), each run included 6 trials (three per condition), and a 10 s fixation at the beginning and end of the run. Each run lasted an average of 184.75 s (SD 3.86 s). Each participant completed 4 runs.

A.5 Defining fROIs

Function regions of interest (fROIs) were defined using the group-constrained subject-specific (GSS) approach [24] where a set of spatial parcels were combined with each individual subjectâĂŹs localizer activation map, to constrain the definition of individual fROIs. The parcels mark the expected gross locations of activations for a given contrast based on prior work and are sufficiently large to encompass the extent of variability in the locations of individual activations.

We reused parcels identified in prior works [23, 24]. For the language localizer, we used a set of six parcels derived from a group-level probabilistic activation overlap map for the sentence reading (SR) > non-word reading (NR) contrast in 220 participants. These parcels included two regions in the left inferior frontal gyrus (LIFG, LIFGorb), one in the left middle frontal gyrus (LMFG), two in the left temporal lobe (LAntTemp and LPostTemp), and one extending into the angular gyrus (LAngG).

For the MD localizer, we used a set of 20 parcels (10 in each hemisphere) derived from a group-level probabilistic activation overlap map for the `hard > easy` spatial task contrast in 197 participants. The parcels included regions in frontal and parietal lobes, as well as a region in anterior cingulate.

Within each parcel, we selected the top 10% most responsive voxels, based on the t-values for the `SR > NR` contrast. Individual-level fROIs defined in this way were then used for subsequent analyses that examined the behavior of the MD and the Language systems during the coding tasks.

A.6 Data processing and analysis

MRI data were analyzed using SPM5. Each participantâĂŹs data were motion corrected and then normalized into a common brain space (the Montreal Neurological Institute (MNI) template) and resampled into 2mm isotropic voxels. The data were then smoothed with a 4mm FWHM Gaussian filter and high-pass filtered (at 200s). Effects were estimated using a General Linear Model (GLM) in which each experimental condition was modeled with a boxcar function (modeling entire blocks) convolved with the canonical hemodynamic response function (HRF).

B ADDITIONAL RESULTS

We provide additional results from our data analysis of the MD and the Language systems.

B.1 fROI responses - MD system

An analysis of activity within individual regions within the MD system showed that 17 of the 20 fROIs in the Python experiment, and 14 of the 20 fROIs in the ScratchJr experiment responded significantly more strongly to code problems than to sentence problems (Figure 5). This demonstrates code processing is broadly distributed across the MD system and is not localized to a particular subset of regions within it.

We evaluate for *selectivity* to *code comprehension* by measuring responses of code problems to hard working memory localizer task for the MD system. Figure 6 plots activations in the various regions of the MD system. We find none for ScratchJr, and three regions in the frontal lobe (precentral-A, precentral-B, midFrontal) which exhibit stronger responses to code problems. However, the magnitude of `code > sent` in these regions ($\Delta\beta$= 1.03, 0.95, 0.97) was comparable to the mean magnitude across all MD system fROIs (average $\Delta\beta$= 1.03), suggesting that the high response was caused by the underlying *code simulation* rather than *code comprehension*.

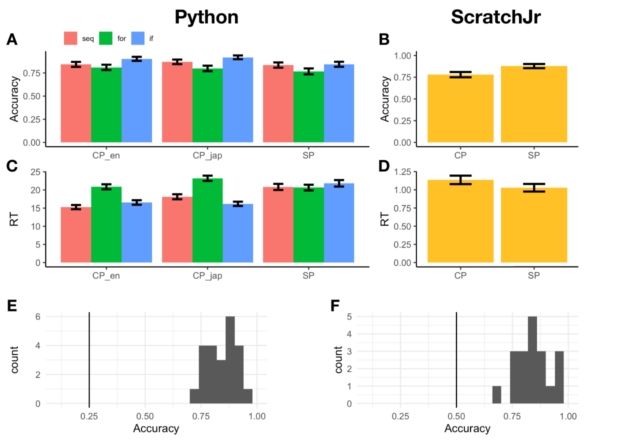


Figure 7: Response accuracies and reaction times to code-related stimuli

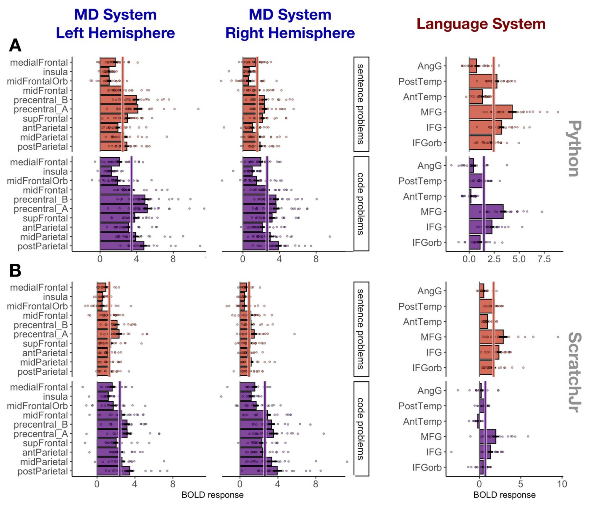


Figure 5: Activations of fROIs to code and sent conditions in the MD system- left hemisphere, MD system- right hemisphere, and the Language system.

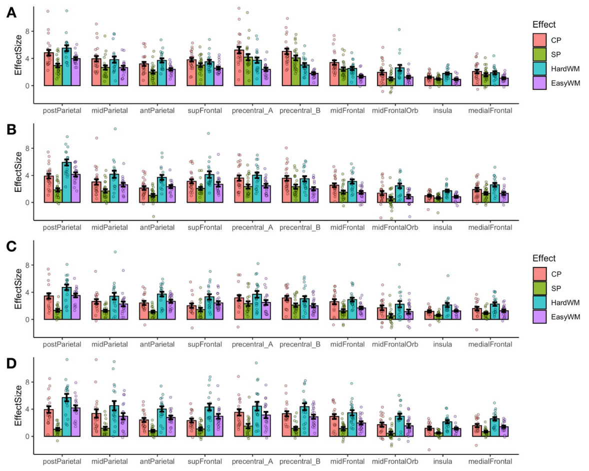


Figure 6: Activations of the MD system fROIs to code and sent conditions, contrasted against the easy and hard MD system localizer tasks. (A) Experiment 1, Python; left hemisphere fROIs; (B) Experiment 1, Python; right hemisphere fROIs; (C) Experiment 2, ScratchJr; left hemisphere fROIs; (D) Experiment 2, ScratchJr; right hemisphere fROIs.

B.2 Behavorial Results

Figure 7 presents response accuracies and reaction times to code-related stimuli presented to our participants in the imaging scanner. Participants in Experiment 1 (Python) had a 99.6% response rate, with an 85% accuracy on average on code problems (Figure 7.A). Figure 7.E shows the histogram of response accuracies of participants in Python.

Participants in Experiment 2 (ScratchJr) had a 98.6% response rate, with 79% accuracy on average on code problems (Figure 7.B). These results demonstrate that participants were proficient in the relevant programming language and engaged with the task. Figure 7.F shows the histogram of response accuracies of participants in ScratchJr.