Response to Comments from Reviewer 2

Thank you for your time and thoughtful comments. I believe that they have helped to greatly improved, not only the manuscript, but the package it describes as well. Please find below a copy of your comments along with my responses.

Comment 2.1

The Introduction immediately introduces the PDP. You must first motivate why it is important. In a linear model, they are not important. In a linear model with simple interactions, likewise. They first become important if you have non-parametric machine-learning techniques. Further, they are not the only idea extant when trying to explain a predictor's impact on the response. This paper would benefit if you can give a bit more of a lit review here.

Response 2.1

I expanded the introduction a bit to try and motivate why PDPs are important (the new material links directly to the Boston housing example used throughout the manuscript). I also included more background on the `plotmo` and `ICEbox` packages (and there corresponding methodologies). `plotmo` very recently introduced a `partdep` option for displaying PDPs. Because of this, more detail was added to differentiate `pdp` and motivate the use of `partial`.

I agree that PDPs are not as useful in "simple" linear models (e.g., few variables, only two-way interactions, etc.), but certainly there will be cases where PDPs will be extremely useful (e.g., many terms, complex interactions, etc.). In fact, linear models are still common in machine learning/computer science/data science (they just typically contain many terms). I also agree that PDPs are not the only, or sometimes best, approach to understanding the impact of a subset of the predictor's on the response. Brief descriptions and references are given for many alternatives: marginal model plots (`plotmo`), individual conditional expectation plots (`ICEbox`)—which are described as an alternatve to PDPs when strong interactions are suspected, effect plots (`effects`), and a few other techniques available in the `car` package.

Important Clarifications Required:

Comment 2.2

Top of p2. The `ICEbox` package allows for general construction and visualization of PDP's. However, it is limited as you only get one dimensional representations. The selling point of your package should focus on the flexibility of graphics and the illustration of non-extrapoloation zones.

Response 2.2

`ICEbox` does not actually construct PDPs, rather individual curves for each observation (which can be difficults to interpret when you have millions of observations). I think flexible graphics and non-extrapoloation zones are only half of the selling point here. I think that being able to run `partial` in parallel is huge, and also that this packge works with many types of machine learning models—not just tree-based models. For example, as described in the manuscript, `pdp` can produce single or multi-predictor PDPs for conditional random forests and XGBoost models (among many others) in parallel. I expanded the last example to describe the interface with `caret`, which I think opens the door for wonderful exploratory data analysis pipe lines (e.g., choose model –> plot importance –> plot relationships, etc.). Also, a new section was added to the manuscript: **User-defined prediction functions**. This section describes the new `pred.fun` argument (as suggested by a later comment) that makes `partial` even more flexible.

Comment 2.3

All `pdp` figures throughout the paper: Can you make explicit in the caption what code produces the left and right subfigures? I would use the subfigure package in latex and then put a comment in the R code e.g. # Fig 3b.

Response 2.3

Rather than using subfigures, which I think would look messy with all the code, I made quite an attempt to comment which pieces of code produced which figure. Also, in response to another reviewer's comments, the code which produces figures containing multiple plots is better described and demonstrated. See, for example, the **Multi-predictor PDPs** section.

Edits:

Comment 2.4

p1 abstract: strict assumptions imposed by traditional statistical models

Response 2.4

Added some clarification here; in particular, normality, constant variance, and linearity.

Comment 2.5

p1 pairs should be using curly braces of angle braces, not square braces as this is easily confused with closed intervals.

Response 2.5

Fixed. Switched to curly braces.

Comment 2.6

p2 unless the editor disagrees, there is no reason to add "install.packages("pdp")" in the text of the paper as it is assumed a user knows how to install a package.

Response 2.6

Fixed. Code chunk was removed.

Comment 2.7

p3 "Error: The training..." should be wrapped.

Response 2.7

Fixed. The text is now wrapped appropriately. Thanks for catching this!

Comment 2.8

Fig 6: the sidebar should be labeled “ozone”. Also, would it be terrible to label all axes even though it is redundant?

Response 2.8

The sidebar (i.e., colorkey) has been labeled, but for this example only. As sad as it sounds, labeling the colorkey is not as trivial as we’d expect for `lattice` plots. The code for doing so has been incorporated in the example in question (see pages 8-9). Also, the text describing the first code chunck that produces a display with a colorkey now contains a footnote pointing to the example where the code to produce a label can be found. This is also motivation for why one of the examples showed how to use `ggplot2` for plotting the results from `partial`—`ggplot2` makes it a trivial matter to label the colorkey. I don’t think it would be terrible to label all the axes. However, in my option, labelling all the axes would (1) looks cluttered, and (2) require more unnecessary code that would probably distract the reader. This is the standard axis labelling scheme used in trellis plots from S-plus to R (via `lattice`).

Comment 2.9

Fig 7: as beautiful as this is, you should label the sidebar as “logit” or “centered logit” instead of “y”.

Response 2.9

Fixed. Color key labeled “Centered logit”.

Suggestions for the software

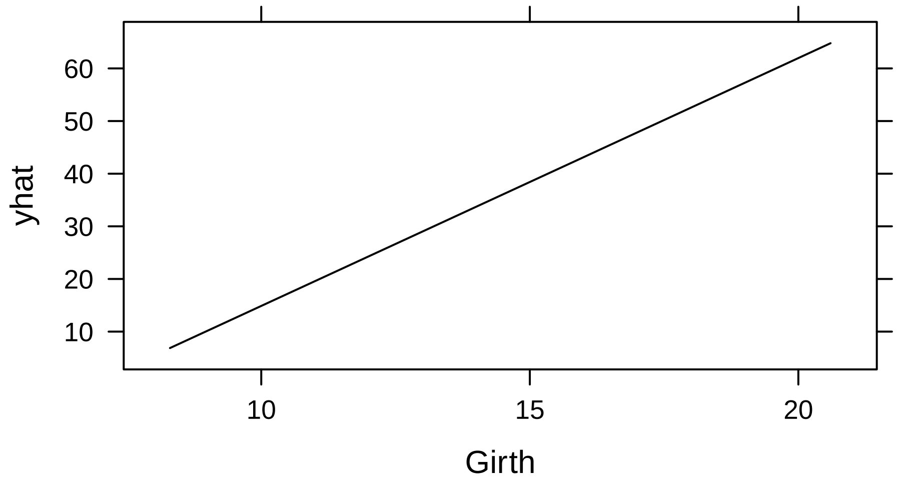
Comment 2.10

I understand that you like the feature of having a one-shot function partial to both generate and plot the PDP. I realize that the default is `FALSE`. However, I still think it’s a disservice to the user as constructing PDP’s is computationally intense and if they get the plot wrong or it’s not to their liking, they will have to regenerate the whole thing again. So, it is my opinion that you should take it out and force two steps: generation and illustration.

Response 2.10

While I am not suggesting that the `partial` function needs to mimic the behavior of the PDP fucntions in `randomForest` and `gbm`, I do find it useful to be able to plot the results without forcing the user to store them first. However, I agree with your concern which is why `partial` has `plot = FALSE` as the default; at the bottom of the second paragraph of section **Constructing PDPs in R** (and in the package documentation), I have added a note empahsizing the use of `plotPartial`. Furthermore, I have made it possible to retrieve the raw PDP data even when `plot = TRUE`. This is because the `"trellis"` object returned by `partial` / `plotPartial` now contains an additional `partial.data` attribute. This is described in the package documentation. For example,

```r
library(pdp)
data(trees)
mod <- lm(Volume ~ ., data = trees)
partial(mod, pred.var = "Girth", plot = TRUE)
```



```r
pdp.Girth <- lattice::trellis.last.object()
pd.Girth <- attr(pdp.Girth, "partial.data")
head(pd.Girth)
```

```
##       Girth      yhat
## 1  8.300000  6.873167
## 2  8.773077  9.100489
## 3  9.246154 11.327811
## 4  9.719231 13.555133
## 5 10.192308 15.782455
## 6 10.665385 18.009777
```

Comment 2.11

The package could be more powerful if you allowed a predict function as a parameter. For instance, you cannot currently use random forests to illustrate classification probabilities.

Response 2.11

This is an excellent suggestion. First, I have changed the behavior of `...` to refer to additional arguments in `predict` (e.g., the `n.trees` argument in `predict.gbm`.). Second, `partial` gained a new argument, `pred.fun`, where the user can provide an optional prediction function. This makes `partial` far more flexible. For example, you can use this argument to construct PDPs on the probability scale, or to construct ICE curves. This is illusrated in the new section labeled **User-defined prediction functions**

Comment 2.12

p7 I believe there are packages allowing for parallelization in Windows. Can you switch the example so this works on all platforms?

Response 2.12

Good suggestion. I tweaked the example to run on both Windows and Unix-like systems. This is also reflected in the manuscript text.