The results below are generated from an R script.

````
---
title: "R Notebook"
output:
  html_document:
    df_print: paged
  html_notebook: default
---

# Running on Discovery

I recommend viewing this with a web-based Rstudio server on Discovery:

https://ood.discovery.neu.edu/pun/sys/dashboard/batch_connect/sys/RStudio/session_contexts/new

Press *Ctrl+Enter* to run a chunk.

# Initialization

*You may want to change the directory below.*

```{r setup}
library(tidyverse)
knitr::opts_knit$set(root.dir = "/scratch/a.guha/minnpm-exp")
```

Load the data:

```{r}
raw_data <- read_csv("results.csv",
  col_types = cols(Status=col_factor(),
                   Project=col_factor(),
                   Rosette=col_logical(),
                   Consistency=col_factor(),
                   Minimize=col_factor(),
                   Time=col_double(),
                   NDeps=col_integer()),
  show_col_types = FALSE)

```


# Manual Verification Step

Check that these are the factors that appear below:

1. *success*: everything worked!
2. *ERESOLVE*: depends on something that isn't in the repository
3. *ETARGET*: requires some other target architecture **verify**
4. *EBADPLATFORM*: requires some other platform (e.g., macOS)
5. *EUNSUPPORTEDPROTOCOL:* a dependency is in a format that NPM does not support
6. *unexpected*: something went wrong on Discovery. See experiment.out
7. *unavailable*: something went wrong and we didn't even capture the result.
````

```
   See experiment.out

```{r}
levels(raw_data$Status)
```

```{r}
levels(raw_data$Consistency)
```

```{r}
levels(raw_data$Minimize)
```

The value in the Count column should be 1000 for every row:

```{r}
raw_data %>%
  group_by(Rosette,Minimize,Consistency) %>%
  summarize(Count = n())
```

Failures

How many failures occur for each configuration? We know we will see more failures than NPM, but hopefull

```{r}
raw_data %>%
  filter(Status != "success") %>%
  group_by(Rosette,Minimize,Consistency) %>%
  summarize(Count = n())
```

Let's rule out failures that are due to unsolvability on our platform:

```{r}
raw_data %>%
  filter(Status == "unexpected" | Status == "unavailable") %>%
  group_by(Rosette,Minimize,Consistency) %>%
  summarize(Count = n())
```

These are likely due to timeouts, Z3 crashing, etc.

The Need for Tree-Solving

NPM uses a tree-solver because its easy to resolve conflicts. But, how many conflicts do you see with Mi

TODO: Need to process output further to distinguish these errors.

Minimizing Number of Dependencies

For each project, the number of dependencies with vanilla NPM, and with MinNPM configured to minimize #
```

```{r}
min_dep_analysis <- bind_rows(raw_data %>%
            filter(Rosette == FALSE) %>%
            select(Project,NDeps) %>%
            mutate(Solver="NPM"),
          raw_data %>%
            filter(Rosette == TRUE & Consistency == "npm" &
                   Minimize == "min_num_deps,min_oldness") %>%
            select(Project, NDeps) %>%
            mutate(Solver="MinDeps")) %>%
  pivot_wider(values_from=NDeps, names_from=Solver)
```

In theory, MinNPM should always produce a smaller solution. But, it seems like it doesn't always.

```{r}
min_dep_analysis %>% mutate(Badness = MinDeps - NPM) %>% filter(Badness > 0)

```

The graph below is bogus, since I've filtered out the outliers.

```{r}
min_dep_analysis %>%
  mutate(Shrinkage = MinDeps / NPM) %>%
  filter(Shrinkage <= 1.0) %>%
  select(Shrinkage) %>%
  ggplot(aes(Shrinkage)) +
  stat_ecdf() +
  ylab("Fraction of solves that shrink")
```

Impact on Priorities

Skip this? Not very informative.

```{r}
bind_rows(
  raw_data %>%
    filter(Rosette == TRUE & Consistency == "npm" &
             Minimize == "min_num_deps,min_oldness") %>%
    select(Project,NDeps) %>%
    mutate(Solver="Deps,Oldness"),
  raw_data %>%
    filter(Rosette == TRUE & Consistency == "npm" &
             Minimize == "min_oldness,min_duplicates") %>%
    select(Project,NDeps) %>%
    mutate(Solver="Oldness,Deps"),
  raw_data %>%
```

```
    filter(Rosette == TRUE & Consistency == "npm" &
             Minimize == "min_duplicates,min_oldness") %>%
    select(Project, NDeps) %>%
    mutate(Solver="Dups,Oldness")) %>%
  ggplot(aes(NDeps,color=Solver)) +
  stat_ecdf()
```

 # Slowdown of MinNPM

These were run on different kinds of machines, etc. and in parallel. So timing i

1. Why is it that the Rosette solver is faster? It seems almost unbelievable that the NPM solver is slow

2. Should we somehow represent the timeouts on this graph?

```{r}
bind_rows(
  raw_data %>%
    filter(Rosette == FALSE) %>%
    select(Project,Time) %>%
    mutate(Solver="NPM"),
  raw_data %>%
    filter(Rosette == TRUE & Consistency == "npm" &
             Minimize == "min_num_deps,min_oldness") %>%
    select(Project,Time) %>%
    mutate(Solver="MinNPM")) %>%
  pivot_wider(values_from=Time, names_from=Solver) %>%
  mutate(DeltaTime = MinNPM- NPM) %>%
  select(-NPM, -MinNPM) %>%
  ggplot(aes(y=DeltaTime)) +
  stat_bin()
```

```
## Error:  <text>:11:3:  unexpected symbol
## 10:
## 11:  I recommend
##       ^
```

The R session information (including the OS info, R version and all packages used):

```
sessionInfo()

## R version 4.0.3 (2020-10-10)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: CentOS Linux 7 (Core)
##
## Matrix products: default
## BLAS:   /shared/centos7/r-project/4.0.3/lib64/R/lib/libRblas.so
## LAPACK: /shared/centos7/r-project/4.0.3/lib64/R/lib/libRlapack.so
##
## locale:
##  [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C               LC_TIME=en_US.UTF-8
##  [4] LC_COLLATE=en_US.UTF-8     LC_MONETARY=en_US.UTF-8    LC_MESSAGES=en_US.UTF-8
```

```
##  [7] LC_PAPER=en_US.UTF-8       LC_NAME=C                  LC_ADDRESS=C
## [10] LC_TELEPHONE=C             LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
## [1] knitr_1.36
##
## loaded via a namespace (and not attached):
## [1] compiler_4.0.3 magrittr_2.0.1 tools_4.0.3    stringi_1.7.5  highr_0.9
## [6] stringr_1.4.0  xfun_0.28      evaluate_0.14
```

```
Sys.time()
```

```
## [1] "2021-12-10 11:50:50 EST"
```