Analysis of a first long training

Daniel Stornetta (dss2q)

Contents

Libraries

```
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.2     v tibble    3.2.1
## v lubridate 1.9.4     v tidyr     1.3.1
## v purrr     1.0.4
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

Loading data

Load in data from an overnight run of the agent on 2025-11-24.

This is from https://github.com/GITHUB-USER-0/SMB_RL/tree/50c9b0ed68e16ba5045b846bdbb408ce94178a41.

Hyperparameters:

- FRAME_WIDTH = 256
- VTRIM = 36
- HTRIM = 36
- HTRIM_RIGHT = 16
- TRIM_FRAME_HEIGHT = 204
- TRIM_FRAME_WIDTH = 204
- ADJ_FRAME_HEIGHT = 100
- ADJ_FRAME_WIDTH = 100
- BUFFER_SIZE = 1000
- SEED = None
- ROM = v0
- BATCH_SIZE = 32
- GAMMA = 0.99
- LEARNING_RATE = 0.0001
- ACTION_SPACE_IN_USE = [[‘right’], [‘NOOP’], [‘right’, ‘B’], [‘right’, ‘A’], [‘A’], [‘down’]]

```
dat = read_csv("./perEpisodeRewards.csv")
```

```
## Rows: 12798 Columns: 4
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (1): course
## dbl (2): episode, cumulativeReward
## lgl (1): flag_get
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

Courses were randomly selected across the entire game.

- episode : training episode
- cumulativeReward : the reward at the end of the episode using the default reward function
- course : the world-stage pairing, eg., "1-1" corresponds to world 1, stage 1, the first course of the game. (For additional details see /documentation/on different SMB courses.ipynb)

```
dat %>% head()
```

```
## # A tibble: 6 x 4
##   episode cumulativeReward course flag_get
##     <dbl>            <dbl> <chr>  <lgl>
## 1       0              324 6-3    FALSE
## 2       1              981 2-3    FALSE
## 3       2             1085 8-1    FALSE
## 4       3              186 8-3    FALSE
## 5       4              192 7-1    FALSE
## 6       5              173 1-4    FALSE
```

A categorization of levels accessed from the Mario Bros. Wiki available under CC BY-SA 3.0 license.

I have added a subcategorization of "Puzzle" for three courses based on their inclusion of looping segments of the course that can repeat until the player runs out of time.

```
courseMap = read_csv("level_categorization.csv") %>%
  mutate(courseType = if_else(course %in% c("4-4", "7-4", "8-4"), paste0(courseType, ", Puzzle"), courseType))
```

```
## Rows: 32 Columns: 2
## -- Column specification --------------------------------------------------------
## Delimiter: ","
```

```
## chr (2): course, courseType
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
courseMap
```

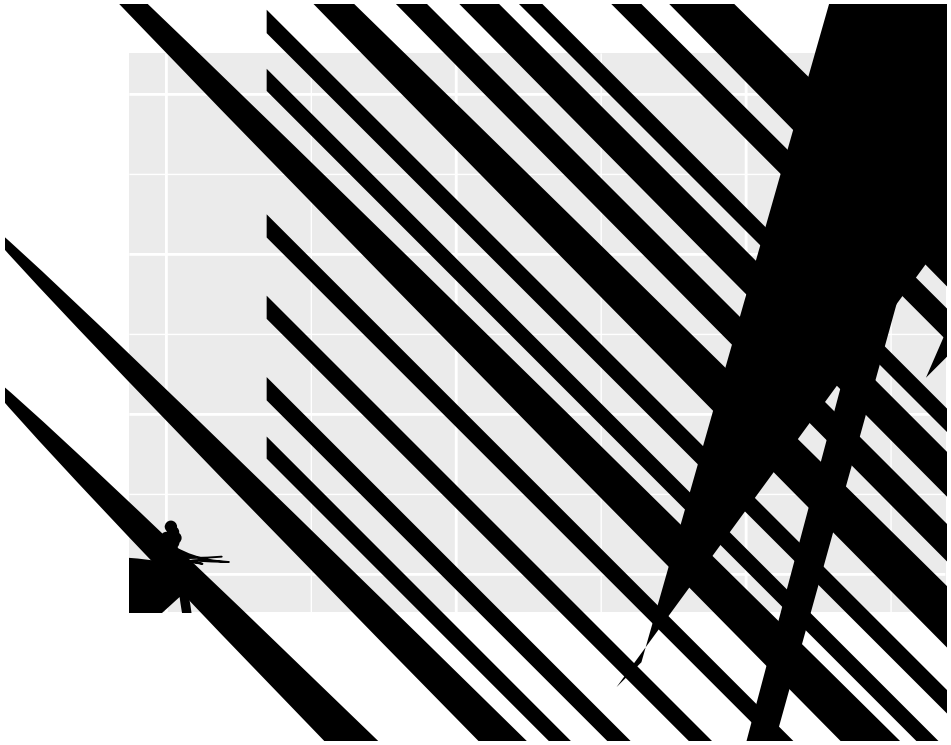
```
## # A tibble: 32 x 2
##    course courseType
##    <chr>  <chr>
##  1 1-1    Ground
##  2 1-2    Underground
##  3 1-3    Athletic
##  4 1-4    Castle
##  5 2-1    Ground
##  6 2-2    Underwater
##  7 2-3    Athletic
##  8 2-4    Castle
##  9 3-1    Ground
## 10 3-2    Ground
## # i 22 more rows
```

Overview of training outcomes

Comments:

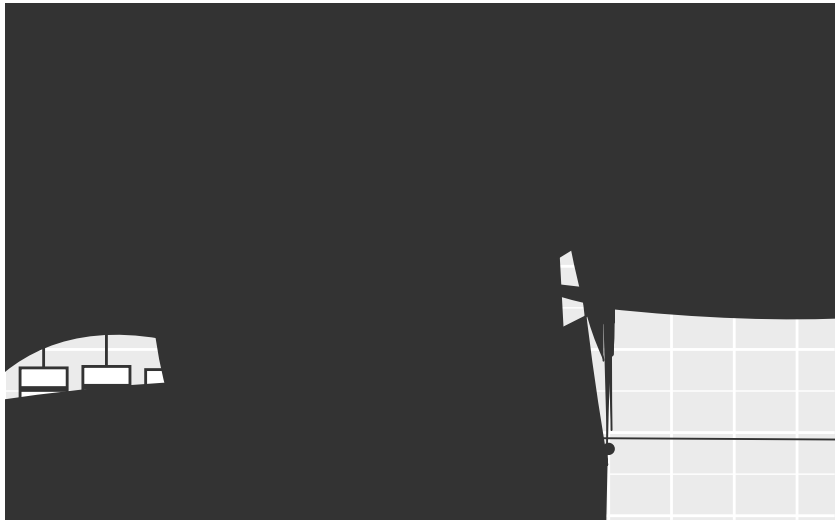
- Outside of sparse high-reward events, it appears that overall rewards were relatively low and stable.
- It appears that there was just one level that managed to reach the flag pole (terminal state for a level).
- Occasional episodes had very high levels of reward. (*This will ultimately align with the hypothesis I had generated about puzzle levels, see 'documentation/on different SMB courses.ipynb'*).

```
episodeCount = nrow(dat)

dat.success = dat %>% filter(flag_get == TRUE)
dat.failure = dat %>% filter(flag_get == FALSE)

ggplot() +
  geom_point(data = dat.failure, aes(x = episode, y = cumulativeReward)) +
  geom_point(data = dat.success, aes(x = episode, y = cumulativeReward), color = "green", size = 2.5) +
  labs(x = "Episode",
       y = "Cumulative Reward (a.u.)",
       title = "Per-episode rewards in Super Mario Bros (1985)",
       caption = paste0("Enlarged green observation reflects the only successfully completed level.",
                        "\nN=", episodeCount))
```



The view of the results when ignoring some of the larger outliers suggests no significant increase in performance over time.

```
dat %>%
  filter(cumulativeReward < 2500) %>%
  mutate(episode_1000 =  floor(episode / 1000)) %>%
  mutate(episode_1000 = factor(episode_1000)) %>%
  ggplot(aes(x = episode_1000, y = cumulativeReward)) +
  geom_boxplot() +
  labs(x = "Episode (1,000s grouped)",
       y = "Cumulative Reward (a.u.)",
       title = "Rewards per 1,000 episodes in Super Mario Bros (1985)",
       subtitle = "Rewards greater than 2,500 not inclusive",
       caption = paste0("Enlarged green observation reflects the only successfully completed level.",
                        "\nN=", episodeCount))
```

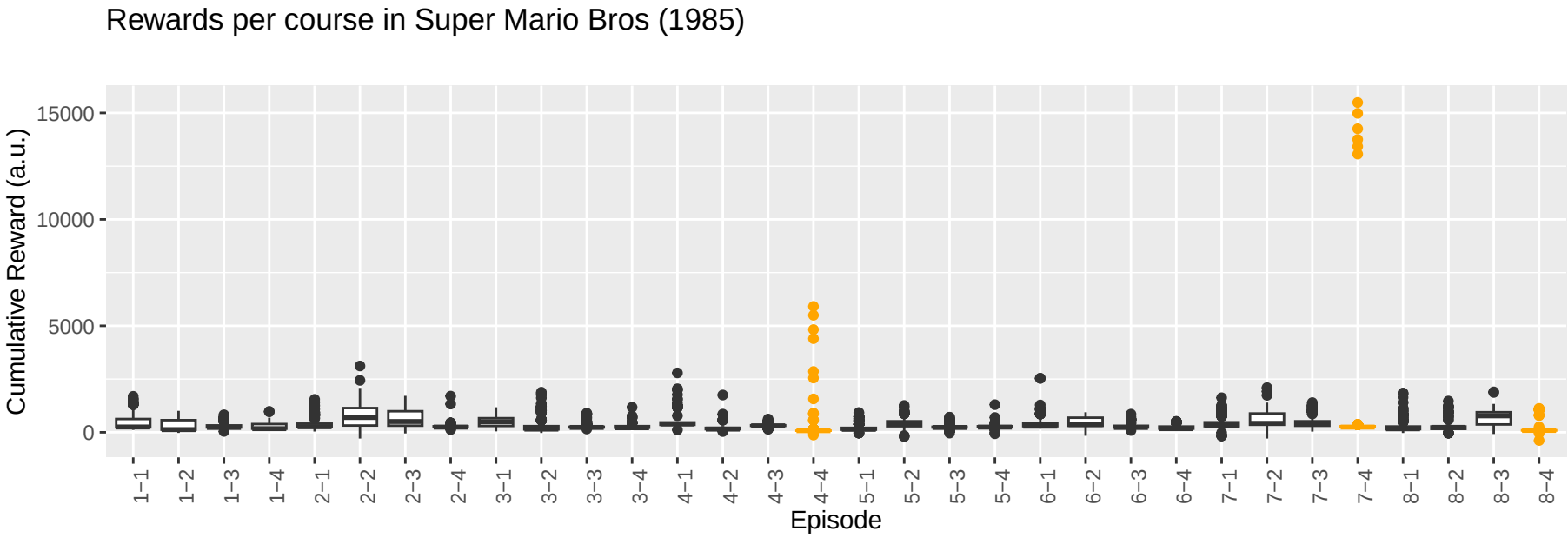


Per-course behavior

A look at per-course behavior provides insight to some of the largest outliers and provides evidence to support a hypothesis generated before testing.

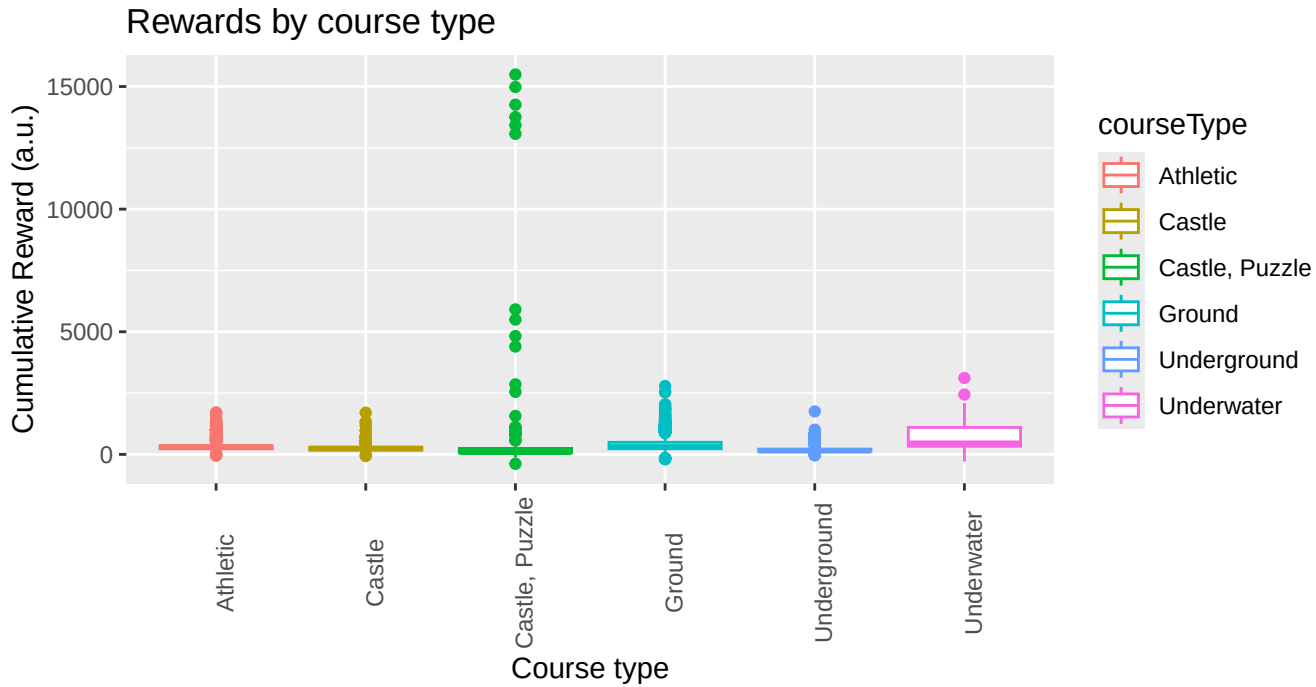
Namely, the reward structure could allow for degenerate reward signaling on 'puzzle levels' that allow for infinitely scrolling courses when the agent takes a looping path, see `/documentation/on different SMB courses.ipynb`.

```
dat.nonPuzzle = dat %>% filter(!course %in% c("4-4", "7-4", "8-4"))
dat.puzzle = dat %>% filter(course %in% c("4-4", "7-4", "8-4"))

ggplot() +
  geom_boxplot(data = dat.nonPuzzle, aes(x = course, y = cumulativeReward)) +
  geom_boxplot(data = dat.puzzle, aes(x = course, y = cumulativeReward), color = "orange") +
  labs(x = "Episode",
       y = "Cumulative Reward (a.u.)",
       title = "Rewards per course in Super Mario Bros (1985)",
       subtitle = "",
       caption = paste0("Episodes with puzzle designs highlighted in orange")) +
  theme(axis.text.x = element_text(angle = 90))
```


Rewards per course in Super Mario Bros (1985)
Cumulative Reward (a.u.)
15000
10000
5000
0
1–1
1–2
1–3
1–4
2–1
2–2
2–3
2–4
3–1
3–2
3–3
3–4
4–1
4–2
4–3
4–4
5–1
5–2
5–3
5–4
6–1
6–2
6–3
6–4
7–1
7–2
7–3
7–4
8–1
8–2
8–3
8–4
Episode
Episodes with puzzle designs highlighted in orange

```
dat %>%
  left_join(courseMap) %>%
  ggplot(aes(x = courseType, y = cumulativeReward, color = courseType)) +
  geom_boxplot() +
  #facet_wrap( ~ courseType, ncol = 2) +
  #coord_cartesian(ylim = c(0, 2500)) +
  theme(axis.text.x = element_text(angle = 90)) +
  labs(x = "Course type",
       y = "Cumulative Reward (a.u.)",
       title = "Rewards by course type",
       caption = paste0("NB., puzzle subcategory added by me, other categorizations as per the Mario Bros Wiki,",
       "CC BY-SA 3.0.\nhttps://www.mariowiki.com/Super_Mario_Bros."))
```

```
## Joining with `by = join_by(course)`
```



Periodic values from saved episodes.

```
dat %>%
  filter(episode %% 1000 == 0) %>%
  knitr::kable()
```

episode	cumulativeReward	course	flag_get
0	324	6-3	FALSE
1000	154	3-2	FALSE
2000	136	3-2	FALSE
3000	166	1-4	FALSE
4000	891	8-3	FALSE
5000	646	2-1	FALSE
6000	1219	4-1	FALSE
7000	949	8-3	FALSE
8000	227	3-3	FALSE
9000	5496	4-4	FALSE
10000	440	7-2	FALSE
11000	373	4-1	FALSE
12000	333	2-3	FALSE

Summary table

Summary table by course.

```
summaryTable = dat %>%
  group_by(course) %>%
  summarize(mean_reward = mean(cumulativeReward),
            sd_reward = sd(cumulativeReward),
            count = n()) %>%
  mutate(mean_reward = round(mean_reward, 2),
         sd_reward = round(sd_reward, 2))

summaryTable %>%
  knitr::kable()
```

course	mean_reward	sd_reward	count
1-1	472.10	328.58	379
1-2	310.91	253.63	381
1-3	275.80	98.17	423
1-4	246.30	132.84	393
2-1	372.98	201.99	390
2-2	734.25	507.59	409
2-3	603.28	358.75	401
2-4	270.85	99.03	416
3-1	477.08	195.37	398
3-2	307.11	317.31	409
3-3	266.63	92.94	389
3-4	252.88	93.35	417
4-1	489.06	331.04	418
4-2	181.59	104.46	401
4-3	319.61	54.55	380
4-4	153.84	537.59	420
5-1	218.03	166.08	408
5-2	411.60	214.51	386
5-3	253.58	101.45	388
5-4	264.73	75.75	392
6-1	364.28	207.22	411
6-2	455.09	233.26	403
6-3	267.12	81.00	393

course	mean_reward	sd_reward	count
6-4	225.27	93.38	427
7-1	391.57	232.65	422
7-2	564.34	381.81	366
7-3	476.61	269.86	385
7-4	465.36	1658.35	418
8-1	291.72	273.46	384
8-2	262.15	209.74	410
8-3	691.15	330.24	400
8-4	116.33	156.68	381

Commentary

Based on a manual review of episodes traces (ie., reviewing saved image frames played together in a movie.)

A manual review of episode 0 suggests that it rapidly is valuing the rightward movement.

A manual review of episode 9,000 provides several observations:

- This is a puzzle level, and the agent is indeed proceeding rightward on a wrong path, allowing it to continue to accumulate rewards.
- The reward function uses the delta in x_position to calculate x_velocity. However, in stages that loop, there are sudden shifts in x_position, jumping from approximately 1,060 (highest observed 1,064) to a thousand less. Thus, this results in single reward steps of 1,000. This could be argued to be a bug of the environment, but also a suggestion that there should be greater reward stabilization/normalization to prevent learning too much from these blips. More simply, the removal of these levels in training may be appropriate.
- – This is supported by how the source code looks at raw RAM values in the emulator.Source code: https://github.com/sajmon83/gymnasium-super-mario-bros/blob/b8f89fbc2da0d495d87b2d6d1e2d56444df73a4b/gym_super_mario_bros/smb_env.py#L139

  ```
  def _x_position(self):
  """Return the current horizontal position."""
  # add the current page 0x6d to the current x
  return int(self.ram[0x6d]) * 0x100 +
         int(self.ram[0x86])
  ```

- There is an additional element of state to the agent, where the previously selected actions limit the behavior of a future action. If the agent is 'holding down' the 'A' button to jump either by itself, or in combination with others, there is 1) a cooldown, and 2) a requirement that the button be released for it to be pressed again. This suggests that the combination of prior actions into a model could be valuable, not just a stacking of prior states.
  - Not that we are seeking to resolve this with dynamic programming, but this information suggests that the Markov assumption is violated if one is considering only the static frame as the state. If momentum values or prior input presses are part of the state then one could restore this property.
  - Regardless of other potential fixes, this also suggests the importance of annealing the epsilon value over time.
- The agent is predominantly holding down ['right', 'A'], which moves it right and jumps. However, by holding down the 'A' button, it is unable to jump unless it releases that button for one or more frames first. The use of an epsilon value of 0.01 appears to be responsible for the jumping observed in this particular episode. Thus, the successful dodging that we observe, may simply be a result of sampling the system many thousands of times and observing successful instances.

Additional comments on 7-4

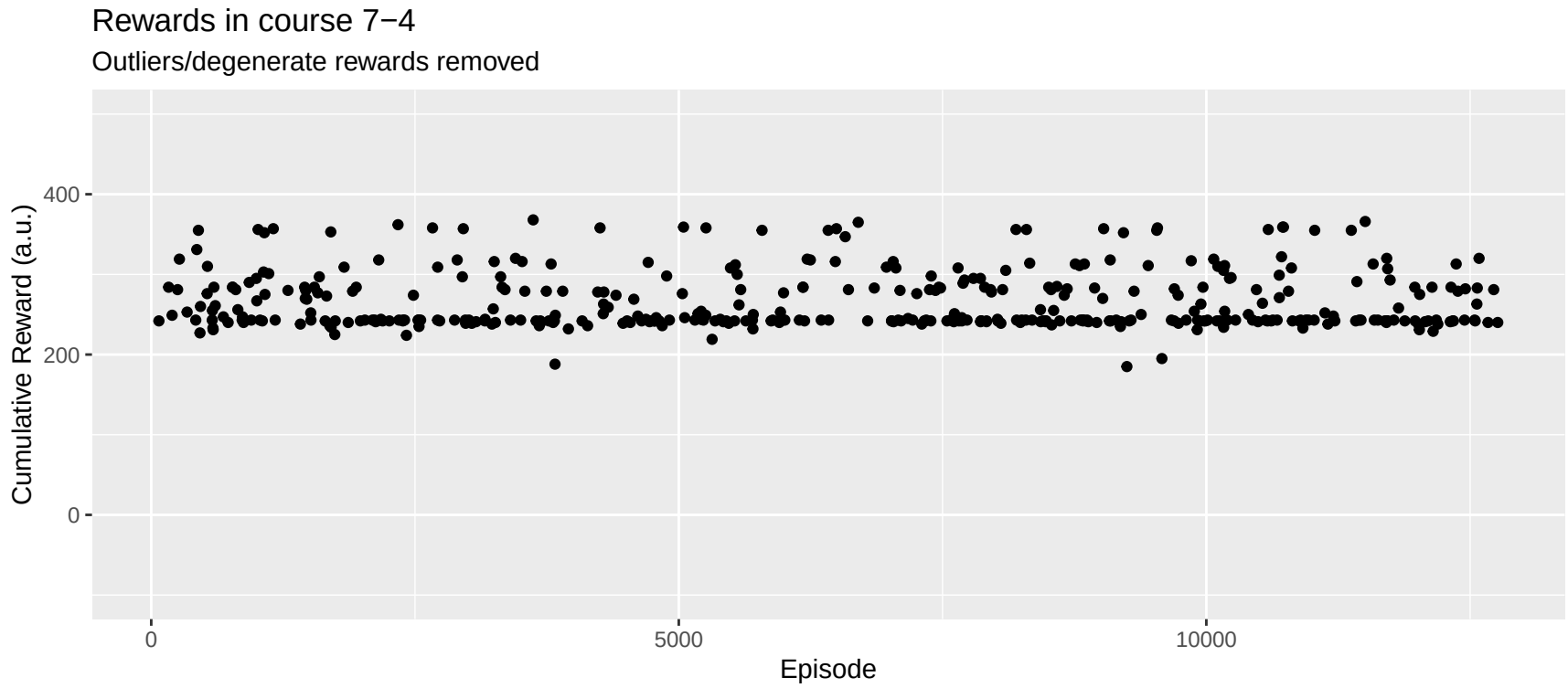
It is interesting to note that there are no instances of negative reward in the course 7-4. This, in spite of hazards being placed near the start of the level. One possible interpretation of this is that the value of the remaining time is counterbalancing the lack of x_position progress, as it seems implausible that the agent is not in fact dying in some instances. This suggests that for post-mortem analysis, one should have the x_position and y_position values stored as well, as they are helpful (notwithstanding the problems of looping x_position levels, which arguably might benefit from being removed.)

It begs the question of whether a reward signal based on the x_position of the agent, relative to the end of the level might be more informative, albeit less generic. Moreover, that would require leaking information about the level from the model to the agent, something that it should not know from the environment.

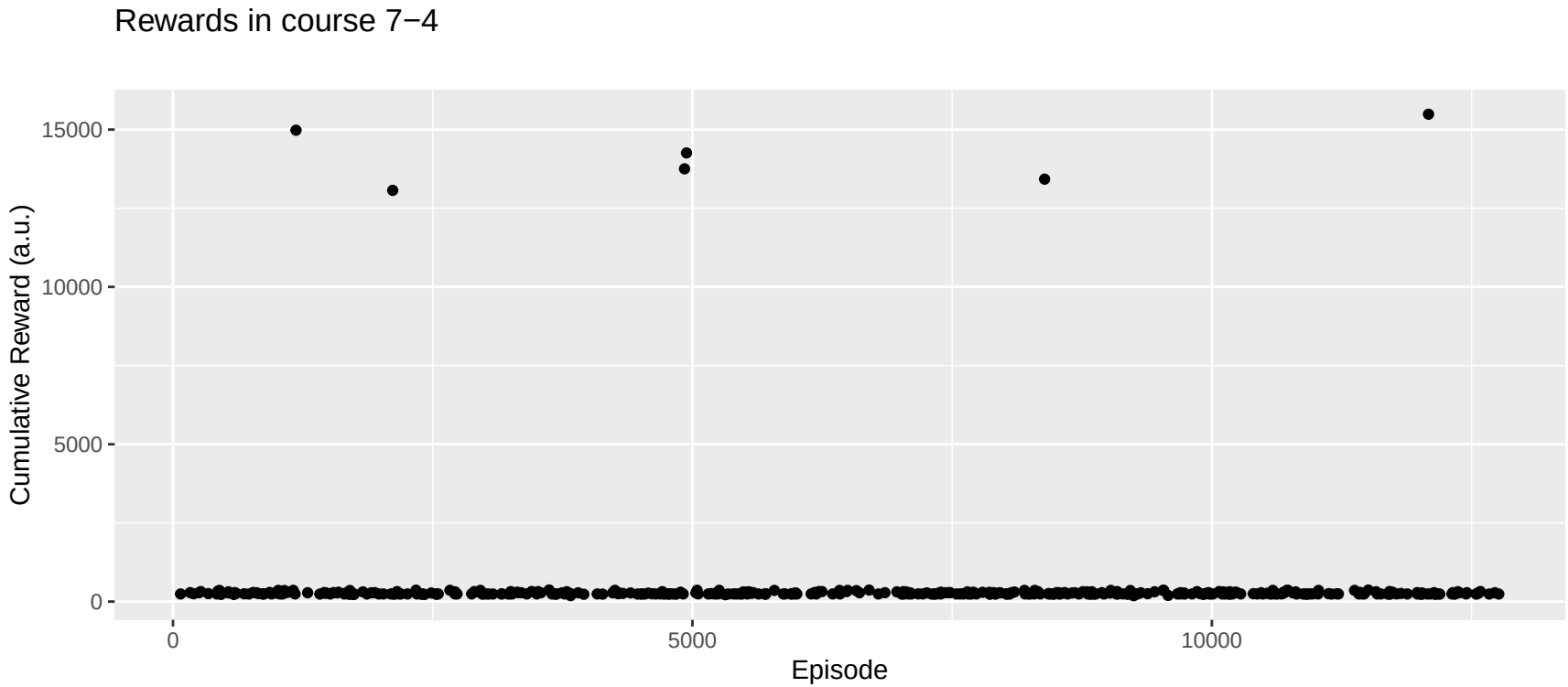
```r
dat %>%
  filter(course == "7-4") %>%
  filter(cumulativeReward < 0)
```

```
## # A tibble: 0 x 4
## # i 4 variables: episode <dbl>, cumulativeReward <dbl>, course <chr>,
## #   flag_get <lgl>
```

```r
dat %>%
  filter(course == "7-4") %>%
  filter(cumulativeReward < 5000) %>%
  ggplot(aes(x = episode, y = cumulativeReward)) +
  geom_point() +
  labs(x = "Episode",
       y = "Cumulative Reward (a.u.)",
       title = "Rewards in course 7-4",
       subtitle = "Outliers/degenerate rewards removed") +
  coord_cartesian(ylim = c(-100, 500))
```




```
dat %>%
  filter(course == "7-4") %>%
  ggplot(aes(x = episode, y = cumulativeReward)) +
  geom_point() +
  labs(x = "Episode",
       y = "Cumulative Reward (a.u.)",
       title = "Rewards in course 7-4",
       subtitle = "")
```


Rewards in course 7−4
Cumulative Reward (a.u.)
15000
10000
5000
0
0
5000
10000
Episode