TL;DR

- There really **isn't** a best day to get married as there isn't much differentiation between various days. Both as part of this analysis and in life. Do what's best for you and your relationship.
- However, if **ALL** you care about is both being married on a Saturday, maximizing the number of times your anniversary will fall on a Saturday, and maximizing the number of "big" anniversaries that fall on a Saturday then avoid the 24 months after a leap day!

The original objective of this analysis

Having somewhat recently celebrated a 5th anniversary on a Saturday (🥂), Mrs. JLaw asked "How many anniversaries will we have on a Saturday?" and "When is the next big one that we'll have?". Upon finding out that our next "big" Saturday anniversary won't come until our 50th, she suggested that I look into whether certain days would have been best to have gotten married.

In actually looking into this analysis, there's not that much difference in the number of Saturdays or "big" Saturdays regardless of wedding date.

So the initial question was, ***what are the BEST and WORST dates to get married*** when optimizing for *maximizing the number of "big" (multiples of 5) anniversaries occurring on a Saturday*. The constraints being that the initial wedding date ALSO needed to be a on a Saturday.

Exploring Wedding Dates and Anniversaries

Since I'll be working with dates the `lubridate` package will be the workhorse for preparing my data.

```
library(tidyverse) #Data Manipulation
library(lubridate) #Working with Dates
library(glue) # A package that works similar to the paste function
library(eulerr) # A package to create Venn-Diagrams (technically Euler
Diagrams)
```

In order to make the universe of wedding dates tractable I'll be looking at all potential dates occurring on a Saturday in the past 10 years (since 1/1/2010) and through the next 5 years (through 12/31/2025). The `seq.Date()` function from the `lubridate` package makes generating sets of dates super easy. It works similar to `seq()` where you give it a starting point, and ending point but in this case you also provide the interval ('day', 'month', 'year', etc.).

In the following code block, I'm constructing a tibble with a column called *wedding_date* that is all days between 1/1/2010 and 12/31/2025 using the `ymd()` function from `lubridate` to turn the integers into a date. Then I'm creating a column called *wedding_date_day* that uses the `wday()` function from `lubridate` to return the day of the week. The "abbr" and "label" options have it return "Mon", "Tue", "Wed" rather than integer values which is the default (this is in part because I constantly forget whether 1 refers to Sunday or Monday… so this eliminates that problem). Finally, I keep only dates that are Saturdays and remove leap days since those will get weird as we look at annual anniversaries.

```
wedding_dates <- tibble(
  wedding_date = seq.Date(ymd(20100101), ymd(20251231), by = 'day'),
  wedding_date_day = wday(wedding_date, abbr = T, label = T)
) %>%
  #Keep only Saturdays
  filter(wedding_date_day == 'Sat') %>%
  #Remove Leap Years because they're unique
  filter(!(day(wedding_date)==29 & month(wedding_date) == 2))
```

This will create a tibble with 834 rows representing all Saturdays between 2010 and 2025.

Counting the Number of Saturday Anniversaires and “Big” Saturday Anniversaries

I will look at the first 50 years of marriage for any of these wedding dates. So for each of the 834 potential wedding dates I need to:

1. Calculate the day of week for each anniversary for the next 50 years
2. For each wedding date, count the number of anniversaries that fall on a Saturday
3. For each wedding date, count the number of “big” anniversaries that fall on a Saturday (again, “big” anniversaries being multiples of 5 such as 5th, 10th, …)

At first I really wanted to figure out a way to do this in a wide-format using `map` functions or `rowwise` functions, but in the end I couldn’t figure it out in the time I wanted to spend exploring. Therefore, I’m keeping the data in a long-format by using `tidyr::crossing()` to expand each wedding days by the 50 anniversaries. So in the end each row in the initial data set will now have 50 rows.

Then for each of the Wedding Date/Anniversary Year combinations, I re-use the `wday()` function to get the day of the week and then `group_by` the wedding date and `summarize()` to count the number of Saturday anniversaries (*num_sat*) and “big” Saturday anniversaries (*num_big_sat*).

The two non-typical parts of this code block are the `.groups` argument to `summarize()` and the use of `paste()` in the `summarize()`. The `.groups` argument returns an ungrouped tibble rather than only removing the last grouping layer which is the default (this would have returned a grouped tibble with *wedding_date* as the grouping variable… which would probably be fine but occasionally grouped tibbles cause downstream issues).

Using `paste()` in the `summarize()` with the `collapse=', '` argument creates a concatenated comma-space separated string of the “big” anniversary years that fall on a Saturday and `NA` otherwise. The use of `stringr::str_remove_all()` is to remove the NAs from the string.

If you’re reading this and are unfamiliar with regular expressions, I highly recommend getting familiar with them, especially when working with text. The regular expression “NA,? ?” means to remove the pattern “NA” followed by 0 or 1 commas followed by 0 or 1 spaces. But the TL;DR here is that when a “big” anniversary didn’t fall on a Saturday the string “NA” would be concatenated and I wanted to remove those. So “5, NA, NA, NA, 45, NA” would just become “5, 45,”. Not ideal.. but it’ll do.

```
wedding_dates_w_annv <- wedding_dates %>%
  #Expand Each Date to Have 50 Anniversaries
  crossing(anniversary = 1:50) %>%
  #Get the Day of Week for Those Anniversaries
  mutate(anniversary_day = wday(wedding_date + years(anniversary),
label = T, abbr = T)) %>%
  #Summarize By Wedding Date counting the number of saturdays, number
of saturdays w/ meaningful anniversary
  group_by(wedding_date, wedding_date_day) %>%
  summarize(
    num_sat = sum(anniversary_day == 'Sat'),
    num_big_sat = sum(anniversary_day == 'Sat' & anniversary %% 5 ==
0),
    #Building a string of all meaningful anniversary years,
    big_sat_years = str_remove_all(
      paste(
        if_else(anniversary_day == 'Sat' & anniversary %% 5 == 0,
                anniversary,
                NA_integer_
                ),
        collapse = ', '),
      "NA,? ?"),
    .groups = 'drop'
  )
```

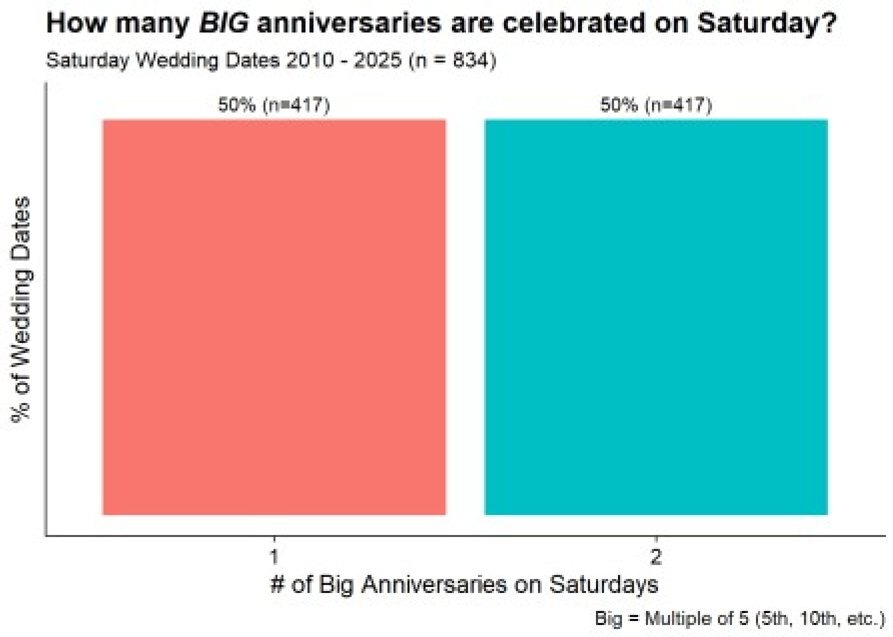
Post-processing the data looks like:

wedding_date	wedding_date_day	num_sat	num_big_sat	big_sat_years
2010-01-02	Sat	7	1	45,
2010-01-09	Sat	7	1	45,
2010-01-16	Sat	7	1	45,

How many “Big” Saturday Anniversaries Does Anyone Get?

The first question to explore is for the 834 Saturdays in our data as potential wedding dates, how many of the “big” anniversaries will fall on a Saturday. The following code block is pretty vanilla `dplyr` with the use of `count()` and `mutate()`. If you’ve never seen the `glue()` package and function before, it works a lot like `paste()` in its most basic form. The main difference is that R will execute the code within the `{ }` so it can be included directly within the quotes rather than separated by commas. It can also be used similar to `.format()` in Python.

```
wedding_dates_w_annv %>%
  # Get frequencies of Big Saturday Anniversaries
  count(num_big_sat) %>%
  # Create %s
  mutate(pct = n/sum(n)) %>%
  ggplot(aes(x = as.factor(num_big_sat), y = pct, fill =
as.factor(num_big_sat))) +
    geom_col() +
    geom_text(aes(label = glue("{pct %>% scales::percent()} (n={n %>%
scales::comma()})")), nudge_y = 0.02) +
    labs(title = "How many ***BIG*** anniversaries are celebrated on
Saturday?",
         subtitle = glue("Saturday Wedding Dates 2010 - 2025 (n =
{nrow(wedding_dates_w_annv)})"),
         caption = "Big = Multiple of 5 (5th, 10th, etc.)",
         x = "# of Big Anniversaries on Saturdays",
         y = "% of Wedding Dates") +
    scale_fill_discrete(guide = F) +
    cowplot::theme_cowplot() +
    theme(
      plot.title = ggtext::element_markdown(),
      axis.text.y = element_blank(),
      axis.ticks.y = element_blank()
    )
```

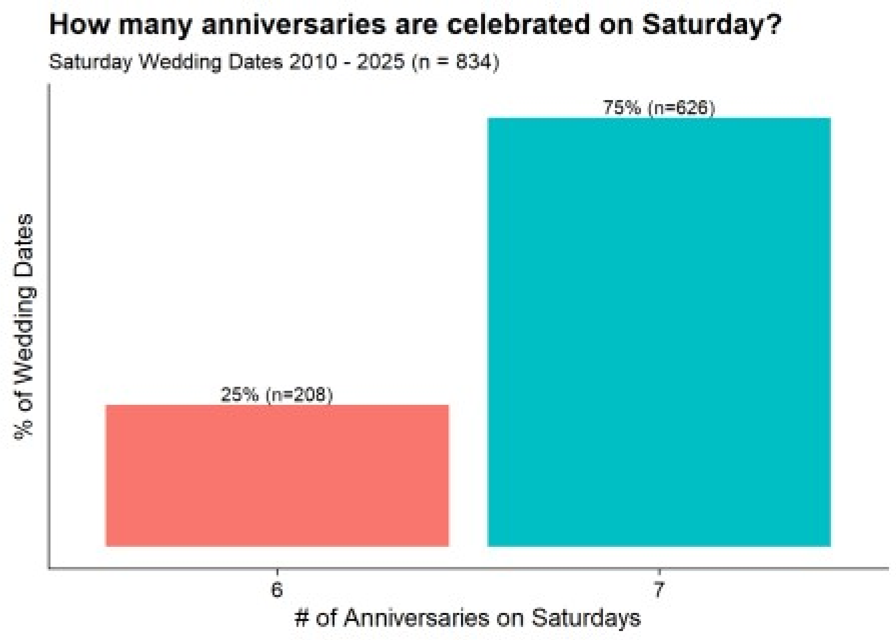


The primary reason there **isn't** a best or worst wedding date is that all potential wedding dates either have 1 or 2 **BIG** anniversaries on a Saturday. So there isn't too much of a difference in choice of dates.

So let's look at how many anniversaries in total occur on a Saturday.

How many total Saturday Anniversaries Does Anyone Get?

```
wedding_dates_w_annv %>%
  count(num_sat) %>%
  mutate(pct = n/sum(n)) %>%
  ggplot(aes(x = as.factor(num_sat), y = pct, fill =
as.factor(num_sat))) +
  geom_col() +
  geom_text(aes(label = glue("{pct %>% scales::percent()} (n={n %>%
scales::comma()})")), nudge_y = 0.02) +
  labs(title = "How many anniversaries are celebrated on Saturday?",
       subtitle = glue("Saturday Wedding Dates 2010 - 2025 (n =
{nrow(wedding_dates_w_annv)})"),
       x = "# of Anniversaries on Saturdays",
       y = "% of Wedding Dates") +
  scale_fill_discrete(guide = F) +
  cowplot::theme_cowplot() +
  theme(
    axis.text.y = element_blank(),
    axis.ticks.y = element_blank()
  )
```

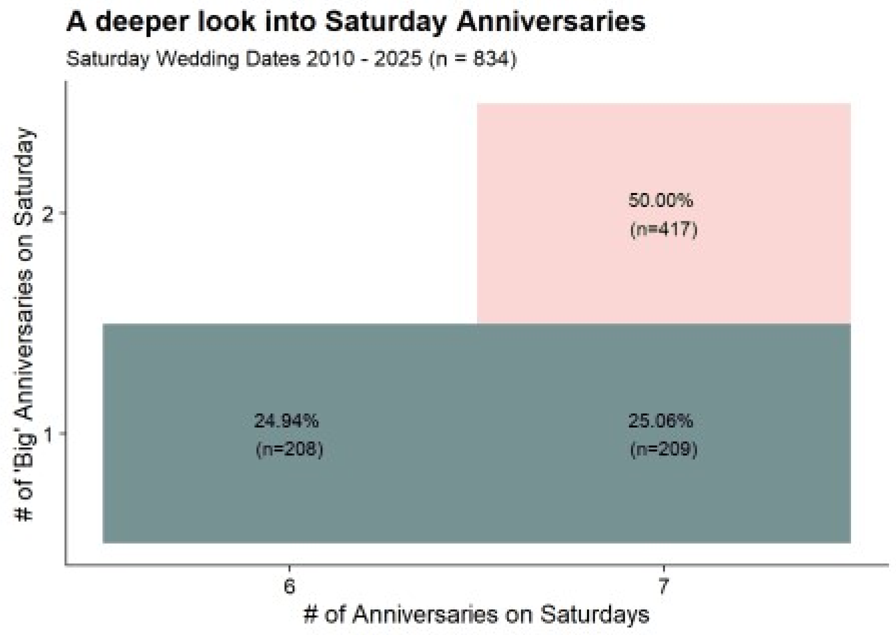


Continuing with the theme of there not being major differences. 75% of Saturday Wedding dates will have 7 anniversaries on a Saturday and 25% will have 6. So, while 7 would be preferable, the difference between 7 vs. 6 again is not large.

Looking at Both Total Saturdays and “Big” Saturdays

Since “Big” Anniversaries had a 50/50 distribution and overall Saturdays had a 25/75 distribution the next step would be to see the cross-product of the two previous fields:

```
wedding_dates_w_annv %>%
  count(num_sat, num_big_sat) %>%
  mutate(pct = n / sum(n)) %>%
  ggplot(aes(x = factor(num_sat), y = factor(num_big_sat), fill = pct))
+
    geom_tile() +
    geom_text(aes(label = glue("{pct %>% scales::percent()} \n
(n={n})"))) +
    labs(title = "A deeper look into Saturday Anniversaries",
         subtitle = glue("Saturday Wedding Dates 2010 - 2025 (n =
{nrow(wedding_dates_w_annv)})"),
         x = "# of Anniversaries on Saturdays",
         y = "# of 'Big' Anniversaries on Saturday") +
    scale_fill_gradient(guide = F, low = "#769293", high = "#fad7d5") +
    cowplot::theme_cowplot()
```



Looking across both dimensions, everyone who has two "big" anniversaries on a Saturday ALSO has 7 anniversaries on a Saturday. However, not everyone who has 7 anniversaries on Saturday will have 2 "big" anniversaries on a Saturday. Instead there are three groups:

- 6 Total / 1 Big (25%)
- 7 Total / 1 Big (25%)
- 7 Total / 2 Big (50%)

In this case, having 7 anniversaries and 2 "Big" Anniversaries seems preferable to the other two groups… if you only cared about having your anniversary on a Saturday.

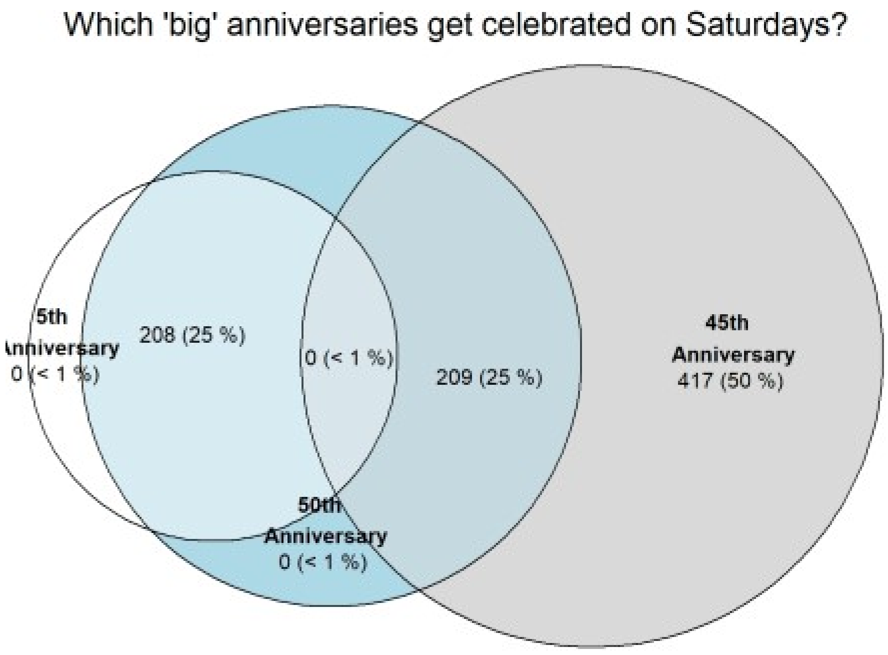
What "Big" Anniversaries Will Be Celebrated on Saturdays?

So far, I've defined "big" anniversaries as multiples of 5 (5th, 10th, … 45th, 50th). However, I haven't looked at which of those big ones are occurring on a Saturday. To show these "big" anniversaries I'll use the `eulerr` package to create a Venn-Diagram of these years.

The package expects a specific format where each column is a logical indicating whether or not an observation is a member of that group. From a quick check on the *big_sat_years* field I can see that the only "big" anniversaries that fall on Saturdays are the 5th, 45th, and 50th.

Of note is the regular expression "\b5\b" for identifying the 5th anniversary. `\\b` represents a word boundary so it is included to make sure that the 5th anniversary doesn't accidentally get picked up by `str_detect()` as part of 45 or 50, which would occur if I only searched for "5".

```
wedding_dates_w_annv %>%
  #Constructing Logicals for Venn Diagrams
  transmute(
    `5th \n Anniversary` = str_detect(big_sat_years, '\\b5\\b'),
    `45th \n Anniversary` = str_detect(big_sat_years, '45'),
    `50th \n Anniversary` = str_detect(big_sat_years, '50')
  ) %>%
  #Plot the Venn-Diagram
  euler() %>%
  plot(quantities = list(type = c('counts', 'percent')),
       percentages = TRUE,
       main = "Which 'big' anniversaries get celebrated on Saturdays?",
       )
```



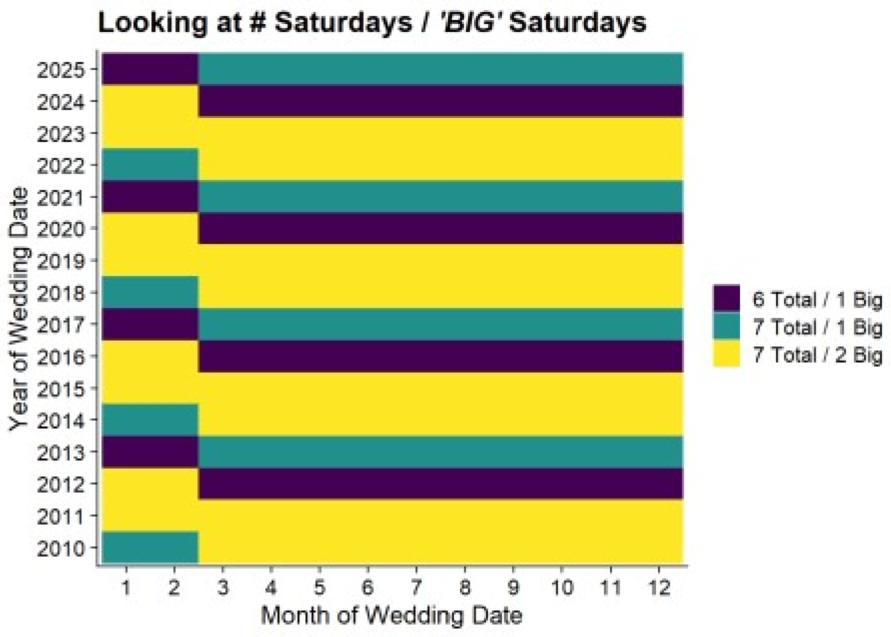
So 75% of wedding dates will celebrate their 45th anniversary on a Saturday. 50% will celebrate ONLY their 45th anniversary and 25% will celebrate their 45th and 50th anniversaries on a Saturday. The last 25% will celebrate their 5th and 50th anniversary on a Saturday. No one will ONLY celebrate either their 5th or 50th. Fitting this into our three group paradigm from the prior section:

- 6 Total / 1 Big (25%) – Will **ONLY** celebrate their 45th Anniversary
- 7 Total / 1 Big (25%) – Will **ONLY** celebrate their 45th Anniversary
- 7 Total / 2 Big (50%)
  - 25% will celebrate their 5th and 50th
  - 25% will celebrate their 45th and 50th

Is there a time component to which group you end up in?

This final section looks at the time component to whether you wind up in the 6/1, 7/1, or 7/2 group. In order to summarize to a Year/Month level, the average number of Saturdays and “Big” Saturdays will be used. Then in the following heat-map, the year of the wedding date appears on the y-axis and the month of the wedding is on the x-axis.

```
wedding_dates_w_annv %>%
  #Reformat to Year-Month (%Y = Year w/ Century, %m = Month as Zero-
Padded Decimal)
  mutate(
    m = month(wedding_date),
    y = year(wedding_date),
  ) %>%
  group_by(m, y) %>%
  #Get Averages
  summarize(across(starts_with('num'), mean), .groups = 'drop') %>%
  mutate(grp = glue("{num_sat} Total / {num_big_sat} Big")) %>%
  ggplot(aes(x = factor(y), y = factor(m), fill = grp)) +
  geom_tile() +
  scale_fill_viridis_d(option = "D") +
  labs(x = "Year of Wedding Date",
       y = "Month of Wedding Date",
       title = "Looking at # Saturdays / ***'BIG'*** Saturdays",
       fill = "") +
  cowplot::theme_cowplot() +
  theme(plot.title = ggtext::element_markdown()) +
  coord_flip()
```



There appears to be a reproducible pattern to which of the three groups you'll wind up in based on the initial wedding date. Probably not surprisingly this occurs in a 4 year cycle.

- 6 Total / 1 Big – Starts in March after a leap year and continues for the next 12 months.
    - Examples: Mar 2012-Feb 2013, Mar 2016-Feb 2017, Mar 2020-Feb 2021
- 7 Total / 1 Big – The following **12** months after the first group
    - Examples: Mar 2013-Feb 2014, Mar 2017-Feb 2018, Mar 2021-Feb 2022
- 7 Total / 2 Big – The following **24** months after the second group
    - Examples: Mar 2014-Feb 2016, Mar 2018-Feb 2020, Mar 2022-Feb 2024

Conclusion

Weddings (or the choice not to have one) are personal decisions for which there is no right or wrong. ***HOWEVER***, if you should choose to require to have your wedding on a Saturday and want to maximize the number of anniversaries you celebrate on Saturday as well as the number the "big" anniversaries celebrated on Saturdays then you'd do well to avoid the 24 months after leap-day.

But the differences between the three groups identified here are pretty small. So while the original question was what are the best and worst days to get married the good answer is that it really doesn't matter!