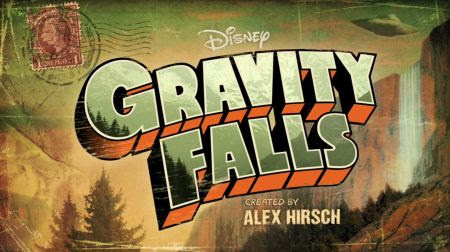
The newest version of {tvthemes} is now on CRAN! v1.1.0 features one new  
palette (Gravity Falls), vignettes, renamed functions.

install.packages("tvthemes") # v1.1.0

library(tvthemes)

**Gravity Falls**

Let’s start off with the main item of interest, the “Gravity Falls”  
palette.



I’ve been meaning to watch Gravity Falls for a while now and once I started I couldn’t stop, it was so much fun! After pretty much binging the show within a week (and reading the extra comic book) I knew I had to make a palette out of the colorful characters!

scales::show\_col(tvthemes:::gravityFalls\_palette)



Altogether what you get is something like this:

library(dplyr)

library(ggplot2)

library(extrafont)

loadfonts(quiet = TRUE)

data <- gapminder::gapminder %>%

filter(country %in% c("Ireland", "Italy", "Turkey", "France", "Germany",

"Brazil", "Mexico", "Sweden", "Netherlands",

"Greece", "Spain", "Finland", "United Kingdom")) %>%

mutate(year = as.Date(paste(year, "-01-01", sep = "", format = '%Y-%b-%d')),

image = "")

ggplot(data = data, aes(x = year, y = gdpPercap, fill = country)) +

geom\_area(alpha = 0.9) +

scale\_x\_date(expand = c(0, 0), breaks = data$year, date\_labels = "%Y") +

scale\_y\_continuous(expand = c(0, 0), labels = scales::dollar) +

scale\_fill\_gravityFalls(reverse = FALSE) +

labs(title = stringr::str\_wrap("Well, Duck-tective, it seems you've really... quacked the case!", width = 70),

subtitle = "Quack-quack Quack-quack-quack (Don't patronize me!)",

caption = "Schmebulock!!",

x = "Years That Stanford Was Gone", y = "# of Sham Total sold") +

theme\_avatar(title.font = "Gravitation Falls",

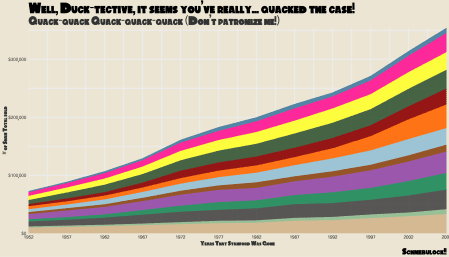
text.font = "Gravitation Falls",

title.size = 24,

subtitle.size = 20,

text.size = 18,

legend.position = "none")



**Renamed functions**

The TV show that changed most was the “Avatar: The Last Airbender” set  
of functions.

* scale\_\*\_avatarTLA() is now: scale\_\*\_avatar()
* theme\_theLastAirbender() is now: theme\_avatar()

Meanwhile a few font functions were renamed to fit the \*\_camelCase()  
style:

* import\_titillium\_web() is now: import\_titilliumWeb()
* import\_roboto\_condensed() is now: import\_robotoCondensed()
* import\_ChelseaMarket() is now: import\_chelseaMarket()

With the exception of import\_ChelseaMarket() all the functions are  
deprecated rather than deleted.

**Vignettes**

Instead of having an abnormally long README on Github I chopped it up  
into a few vignettes:

To create one you need to:

usethis::use\_vignette(name = "name\_of\_vignette\_file",

title = "title\_of\_vignette")

…and start typing away!

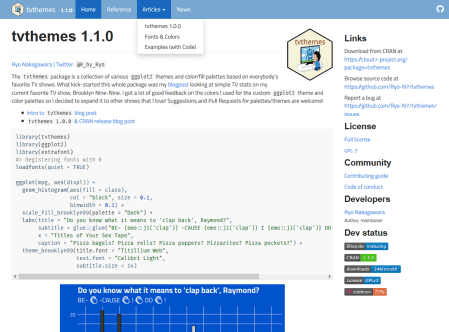
**Creating a pkgdown website**

Along with the vignettes I also created a pkgdown website to house all  
of the documentation. Using {pkgdown} is an extremely easy process which  
mainly takes two functions to get up and running (this is assuming you  
have all your documentation written up of course).

usethis::use\_pkgdown()

pkgdown::build\_site()

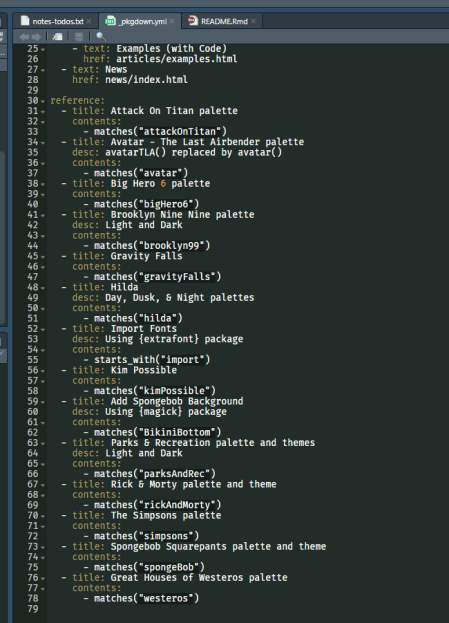
You do need to go into your Github repo settings and set the “Github  
pages” source to be set to your ‘master branch /docs’ folder but for  
actual R related code the above is all you really need!



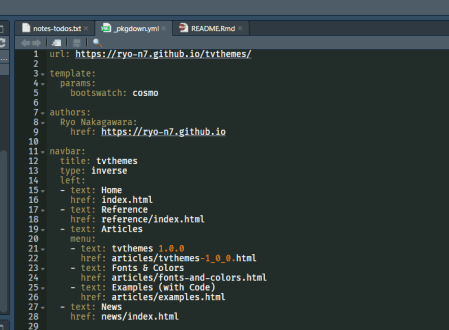
A nice template is already set up based on the documentation you have  
already with:

* Home: Taken from your README.md with extra package info (badges,  
  licenses, download links, etc.) on a sidebar to the right
* Reference: Houses all your function documentation
* Articles: Houses your vignettes/articles
* Changelog: Shows your NEWS.md

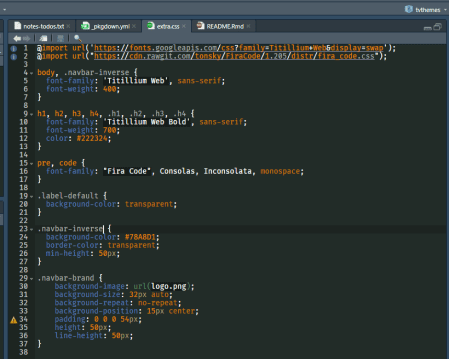
To optimize your website you can fiddle with the \_pkgdown.yml and  
extra.css files. You can set a number of things in the pkgdown YAML  
file such as setting the exact URL of the website, assigning a CSS  
template, and even reorganizing the contents in your pages. The last  
part is especially helpful for {tvthemes} where a lot of functions need  
to be grouped up in the documentation for clarity. You can set up each  
group with a title and then making sure the contents match up using  
the matches(), starts\_with(), group of functions which you might be  
familiar with from {dplyr}.



You can also configure the contents of the navigation bar along the top  
from the YAML file. You can add a title to each item as well as order  
the items in the bar and any drop-down menus in any way you like.



You can add these by specifying  
the stylesheet and theme in the YAML file. Have a look around at other  
packages’ \_pkgdown.yml file to see what kind of stylesheets they use  
as that’s how I found Bootswatch. You can also include an extra.css  
file inside your /docs folder to customize your website template even  
further.



The font I used for the code throughout the website are the “Fira Code”  
ligatures which are the same ones that I use in RStudio, I highly  
recommend it. It gives you nice symbols for “not equals”, assignment arrows, and other symbols used when programming in R which makes it a lot easier to distinguish them from regular code. For the fonts used in the text of the website itself, I used, Titillium Web, which is a font that I love and can also be accessed from within {tvthemes} as well. For free fonts you need not look further than the [Google Fonts](https://fonts.google.com/) or the [Adobe Fonts](https://fonts.adobe.com/fonts) data bases and you can insert them into the extra.css file as shown above.

As described above it can be easy to customize your website and it definintely helps that there are lots of package maintainers with great CSS skills that you can learn from by taking a peek at their Github repos. I don’t know much about CSS but even I managed to create a simple but elegant website after some trial-and-error!

**Conclusion**

This version included only one new palette as it was largely focused on  
correcting previous mistakes and improving the documentation  
significantly with the creation of the package website. I am hoping for  
the next few releases to have more palettes and themes.