



Making R packages

0. Install package creator and documentation tools

1. Create a package
2. Adding files to the package
3. Documenting packages
4. Making the package available to others
5. Creating a Github repository for the package



This page is based on the blog: <https://tinyheero.github.io/jekyll/update/2015/07/26/making-your-first-R-package.html>

0. Install package creator and documentation tools

```
install.packages("devtools")
install.packages("roxygen2")
```

1. Create a package

- A package name `regkurs` is the **created** with the command

```
devtools::create("regkurs")
```

- The create command sets up a **folder structure** for the package:
 - **DESCRIPTION**: This is where all the meta-data about your package goes.
 - **regkurs.Rproj**: This is a RStudio specific file that treats the package as project in RStudio. Can be convenient. Just click on the file in your file explorer and the whole project for the package will be loaded in Rstudio.
 - **R**: This is where all your R code goes for your package. Your source files (.R) with all the functions in the package should be placed here.
 - the **man** folder will contain the documentation, when that is created. This folder will not be available at this stage. See below on creating documentation.
 - the **vignette** folder with the tutorials for the package, if and when such are created.
 - the **data** folder will contain the datasets included in the package, when datasets have been added. This folder will not be available at this stage. See below on adding data.

2. Adding files to the package

- Add source files (.R) to the /R folder. Set the working directory using for example `setwd()` to the folder of the generated package. The following command will build the package and install it on your local machine in the R system library:

```
devtools::install()
```

- The **basic workflow** after a package has been initialized:
 - `devtools::document()` builds the documentation files on your local machine.
 - `devtools::install()`
 - **Including datasets in the package.** Doing `devtools::use_data(mydata)` makes the dataset in `mydata` available in the package as the file `/data/mydata.rda`
- UPDATE: I had to do the following for this to work (on Linux at least):

```
library(devtools)
use_data(mydata)
```

- **Using other packages in your package.** *Never use `library()` in your package.* Instead to this:
 - Add the following in your DESCRIPTION file (tells the package that it depends on these two packages. In the example below, the first package needs to be version 1.9.4 or higher, the other the most recent version):

```
Imports:
  data.table (>= 1.9.4),
  dplyr
```

- use the double-colon `::` when using functions from other packages from within your package. For example `dplyr::filter()` to use the `filter` function from the `dplyr` package.

3. Documenting packages

Functions and datasets in the package can be documented so that users can get help within R with the usual `?`.

- The documentation of a function should be placed right above the function definition for the system to find it when building the docs. Here is an example:

```
#' Summarize the results from a logistic regression analysis
#'
#' Alternative to `summary.glm` to summarize a regression from `glm`.
#' Prints a table similar to the one generated by SAS and Minitab.
#' @param glmobject a fitted regression model from `glm`.
#' @param param `TRUE` if parameter estimates, standard errors etc is computed.
#' @param conf_intervals `TRUE` if confidence intervals for parameters.
#' @param vif_factors `TRUE` if variance inflation factors are to be printed.
#' @return list with two tables: param, odds_ratio
#' @export
#' @examples
#' library(regkors)
#' glmfit <- glm(survived ~ age + sex + firstclass, data = titanic, family = binomial)
#' logisticregsummary(glmfit)
```

R: Summarize the results from a logistic regression analysis - Find in Topic

logisticregsummary {regkurs} R Documentation

Summarize the results from a logistic regression analysis

Description

Alternative to `summary.glm` to summarize a regression from `glm`. Prints a table similar to the one generated by SAS and Minitab.

Usage

```
logisticregsummary(
  glmobject,
  odds_ratio = T,
  param = T,
  conf_intervals = F,
  vif_factors = F
)
```

Arguments

`glmobject` a fitted regression model from `lm`.

`param` TRUE if parameter estimates, standard errors etc is computed.

`conf_intervals` TRUE if confidence intervals for parameters.

`vif_factors` TRUE if variance inflation factors are to be printed.

Value

list with three tables: `param`, `anova` and `fit_measures`

Examples

```
library(regkurs)
simdata <- logisticregsimulate(n = 500, betavect = c(1, -2, 1, 0))
glmfit <- glm(y ~ X, data = simdata, family = binomial)
logisticregsummary(glmfit)
```

[Package *regkurs* version 0.0.1 [Index](#)]

The `@export` makes sure that this function is included in the docs.

- To **build the documentation**, use the command

```
devtools::document()
```

- **Math in documentation.** The equation $y = \beta_0 + \beta_1 x_1 + \dots, \beta_k x_k + \varepsilon$ can be included in the documentation as 'display equation' (use `\eqn{}` for inline equations):

```
\deqn{y = \beta_1 x_1 + \ldots + \beta_k x_k + \epsilon}{y = \beta_1 *x_1 + .... + \beta_k x_k + \epsilon}
```

Note that the maths is written twice:

- first in a latex version which prints well in the pdf
- then in a ascii version which prints well inside R

The general syntax for maths in the docs is hence `\deqn{latex}{ascii}`. It is ok to just use `\deqn{ascii}`, but that will not give you latex in the pdf file of the docs.

- A **dataset** can be **documented** by editing adding an entry in the files in `/R/datasets.R` (you have to create this file once for the package). Here is an example:

```
## Survival of passengers on the Titanic
##
## This data set provides information on the fate of passengers on the fatal maiden voyage of the ocean liner 'Titanic', summarized as
## NOTE: this is not the same as the dataset Titanic (note capital T) which has more observations, but also missing values.
##
## The sinking of the Titanic is a famous event, and new books are still being published about it. Many well-known facts—from the prop
## These data were originally collected by the British Board of Trade in their investigation of the sinking. Note that there is not co
## Due in particular to the very successful film 'Titanic', the last years saw a rise in public interest in the Titanic. Very detailed
##
## @format A data frame with 887 rows and 8 variables:
## \describe{
##   \item{name}{passenger name}
```

```
#' \item{survived}{0 = no, 1 = yes}
#' \item{sex}{male/female}
#' \item{age}{age of passenger}
#' \item{fare}{ticket cost}
#' \item{firstclass}{first class ticket}
#' ...
#' }
#' @source Dawson, Robert J. MacG. (1995), The 'Unusual Episode' Data Revisited. Journal of Statistics Education, 3. doi: 10.1080/1069
"titanic"
```

- The **pdf of the manual** can be build with (may not work on windows)

```
devtools::build_manual(path = "./man") # Builds the pdf in the /man subfolder
```

- **Vignettes** give a nice tutorial style intro to your package:

```
usethat::use_vignette("introduction")
```




This will create a `vignette/introduction.Rmd` file. This is a vignette template Rmarkdown file that you can then use to fill out steps on how you can use your package.

4. Making the package available to others

- If you have a mature package, you can [register it on CRAN](#).
- An easier way to distribute your package is to host it on [GitHub](#). See Point 5 below for info.
- **Users can install the package** with the commands:

```
install.packages("remotes") # install only once to get the install_git() function
library(remotes)
install_git("mattiasvillani/regkurs") # The repository regkurs on my github 'mattiasvillani'
library(regkurs)
```



- Here is the regkurs package on github: <https://github.com/mattiasvillani/regkurs>











 master ▾
  1 branch
  0 tags


Go to file

Add file ▾

Code ▾

 **mattiasvillani** fixed a bug and compiled a pdf manual 819e9fe 10 days ago  17 commits

 R	fixed a bug and compiled a pdf manual	10 days ago
 data	titanic data, logistic regressikon	12 days ago
 man	fixed a bug and compiled a pdf manual	10 days ago
 scripts	adding the sources files after corrupt .gitignore	14 days ago
 vignettes	first commit	14 days ago
 .Rbuildignore	update gitignore	14 days ago
 .gitignore	update gitignore	14 days ago
 DESCRIPTION	missing comma added	12 days ago
 NAMESPACE	titanic data, logistic regressikon	12 days ago
 README.md	fixed a bug and compiled a pdf manual	10 days ago


README.md 

R-package for the course Regressions- och tidsserieanalys 7.5 hp

A package specifically designed for a very first course in Regression and Time Series Analysis using R. Contains functions to summarize the output from regression (lm and glm) in a format closer to that of SAS and Minitab. The idea is to help the transitioning courses to R. The package also contains functions to simulate from regression and logistic regression. More will come.

5. Creating a Github repository for the package

- **Make a github account.** It free for unlimited public repositories. But even private repositories are free if you are an academic: <https://docs.github.com/en/education/explore-the-benefits-of-teaching-and-learning-with-github-education/use-github-in-your-classroom-and-research/apply-for-an-educator-or-researcher-discount>
- Click the 'New repository' button on the top left:


Department of Statistics, Stockholm University

[Overview](#)
[Repositories](#)
[Packages](#)
[People](#)
[Teams](#)
[Projects](#)
[Settings](#)

Type ▾






Language ▾

Sort ▾

New repository






SUdatasets
Public

Datasets used in the teaching at Department of Statistics, Stockholm University






 0
  0
  0
  0
 Updated 11 hours ago

regkurs
Public

R package for the course Regressions- och tidsserieanalys at Stockholm University


 0
  0
  0
  0
 Updated 16 hours ago

.github
Public


 0
  0
  0
  0
 Updated 16 hours ago

- Fill in the info about the repo. Don't click the last three boxes if you are importing an existing repository (which is often what you will do).

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Owner *

 StatisticsSU ▾

Repository name *

/ myprettypackage ✓

Great repository names are short and memorable. Need inspiration? How about **fluffy-chainsaw**?

Description (optional)

This is the beginning of a beautiful repo.

☒  **Public**

Anyone on the internet can see this repository. You choose who can commit.

☐  **Private**

You choose who can see and commit to this repository.

Initialize this repository with:

Skip this step if you're importing an existing repository.

☐ **Add a README file**

This is where you can write a long description for your project. [Learn more.](#)

☐ **Add .gitignore**

Choose which files not to track from a list of templates. [Learn more.](#)

☐ **Choose a license**

A license tells others what they can and can't do with your code. [Learn more.](#)

Create repository

- The above creates an empty repository at Github. Now it is time to make Git repository on your local computer. In the terminal: go to the directory of your R package and type this code in the terminal (assuming your repo is called

`myprettpackage`)

```
echo "# myprettpackage" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
```

You now have a local git repository on your computer (given by the hidden .git folder in your R package directory)!

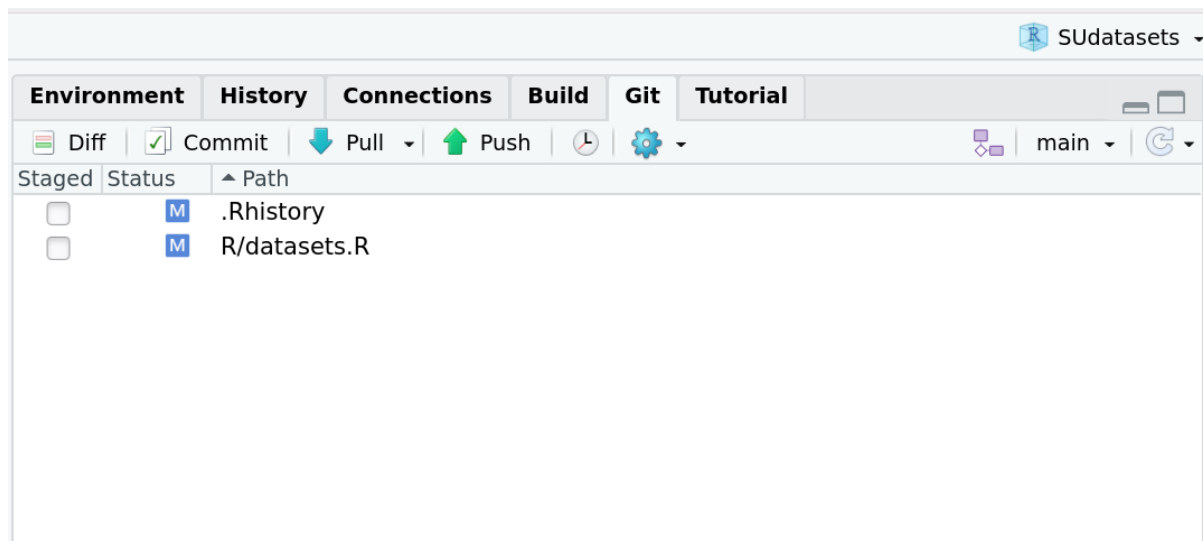
- The final step is to connect your local repository to the one on Github:

```
git remote add origin git@github.com:StatisticsSU/myprettpackage.git
```

- You can now **push** the files in your local repo to the Github repo.

```
git push -u origin main
```

- Or you can **push directly from RStudio**



- You can **pull** someone else's changes in your `myprettypackage` on Github to your local git repository.