Topology Inference for RDF

Jie Xu

2020-12-10

Contents

Chapter 1

Introduction

This website hosts

Chapter 2

Bus and Edge

```r
dat <- tibble::tibble(
  definition = c('transport power from one place to another')
)
print(data)
```

```
## function (..., list = character(), package = NULL, lib.loc = NULL,
##     verbose = getOption("verbose"), envir = .GlobalEnv, overwrite = TRUE)
## {
##     fileExt <- function(x) {
##         db <- grepl("\\.[^.]+\\.(gz|bz2|xz)$", x)
##         ans <- sub(".*\\.", "", x)
##         ans[db] <- sub(".*\\.([^.]+\\.)(gz|bz2|xz)$", "\\1\\2",
##             x[db])
##         ans
##     }
##     my_read_table <- function(...) {
##         lcc <- Sys.getlocale("LC_COLLATE")
##         on.exit(Sys.setlocale("LC_COLLATE", lcc))
##         Sys.setlocale("LC_COLLATE", "C")
##         read.table(...)
##     }
##     names <- c(as.character(substitute(list(...))[-1L]), list)
##     if (!is.null(package)) {
##         if (!is.character(package))
##             stop("'package' must be a character string or NULL")
##         if (FALSE) {
##             if (any(package %in% "base"))
##                 warning("datasets have been moved from package 'base' to package 'datasets'")
##             if (any(package %in% "stats"))
##                 warning("datasets have been moved from package 'stats' to package 'd
##             package[package %in% c("base", "stats")] <- "datasets"
##         }
##     }
##     paths <- find.package(package, lib.loc, verbose = verbose)
##     if (is.null(lib.loc))
##         paths <- c(path.package(package, TRUE), if (!length(package)) getwd(),
##             paths)
##     paths <- unique(normalizePath(paths[file.exists(paths)]))
##     paths <- paths[dir.exists(file.path(paths, "data"))]
##     dataExts <- tools:::.make_file_exts("data")
##     if (length(names) == 0L) {
##         db <- matrix(character(), nrow = 0L, ncol = 4L)
##         for (path in paths) {
##             entries <- NULL
##             packageName <- if (file_test("-f", file.path(path,
##                 "DESCRIPTION")))
##                 basename(path)
##             else "."
##             if (file_test("-f", INDEX <- file.path(path, "Meta",
##                 "data.rds"))) {
##                 entries <- readRDS(INDEX)
##             }
##             else {
##                 dataDir <- file.path(path, "data")
##                 entries <- tools::list_files_with_type(dataDir,
##                   "data")
##                 if (length(entries)) {
##                   entries <- unique(tools::file_path_sans_ext(basename(entries)))
##                   entries <- cbind(entries, "")
##                 }
##             }
##             if (NROW(entries)) {
##                 if (is.matrix(entries) && ncol(entries) == 2L)
##                   db <- rbind(db, cbind(packageName, dirname(path),
##                     entries))
##                 else warning(gettextf("data index for package %s is invalid and will
##                   sQuote(packageName)), domain = NA, call. = FALSE)
##             }
##         }
##         colnames(db) <- c("Package", "LibPath", "Item", "Title")
##         footer <- if (missing(package))
##             paste0("Use ", sQuote(paste("data(package =", ".packages(all.available =
##                 "\n", "to list the data sets in all *available* packages.")
##         else NULL
##         y <- list(title = "Data sets", header = NULL, results = db,
##              footer = footer)
##          class(y) <- "packageIQR"
##          return(y)
##      }
##      paths <- file.path(paths, "data")
##      for (name in names) {
##          found <- FALSE
##          for (p in paths) {
##              tmp_env <- if (overwrite)
##                  envir
##              else new.env()
##              if (file_test("-f", file.path(p, "Rdata.rds"))) {
##                  rds <- readRDS(file.path(p, "Rdata.rds"))
##                  if (name %in% names(rds)) {
##                    found <- TRUE
##                    if (verbose)
##                      message(sprintf("name=%s:\t found in Rdata.rds",
##                        name), domain = NA)
##                    thispkg <- sub(".*/([^/]*)/data$", "\\1", p)
##                    thispkg <- sub("_.*$", "", thispkg)
##                    thispkg <- paste0("package:", thispkg)
##                    objs <- rds[[name]]
##                    lazyLoad(file.path(p, "Rdata"), envir = tmp_env,
##                      filter = function(x) x %in% objs)
##                    break
##                  }
##                  else if (verbose)
##                    message(sprintf("name=%s:\t NOT found in names() of Rdata.rds, i.e.,\n\t%s\n
##                      name, paste(names(rds), collapse = ",")),
##                      domain = NA)
##              }
##              if (file_test("-f", file.path(p, "Rdata.zip"))) {
##                  warning("zipped data found for package ", sQuote(basename(dirname(p))),
##                    ".\nThat is defunct, so please re-install the package.",
##                    domain = NA)
##                  if (file_test("-f", fp <- file.path(p, "filelist")))
##                    files <- file.path(p, scan(fp, what = "", quiet = TRUE))
##                  else {
##                    warning(gettextf("file 'filelist' is missing for directory %s",
##                      sQuote(p)), domain = NA)
##                    next
##                  }
##              }
##              else {
##                  files <- list.files(p, full.names = TRUE)
##              }
##              files <- files[grep(name, files, fixed = TRUE)]
##              if (length(files) > 1L) {
##                  o <- match(fileExt(files), dataExts, nomatch = 100L)
##                  paths0 <- dirname(files)
##                  paths0 <- factor(paths0, levels = unique(paths0))
##                  files <- files[order(paths0, o)]
##              }
##              if (length(files)) {
##                  for (file in files) {
##                    if (verbose)
##                      message("name=", name, ":\t file= ...", .Platform$file.sep,
##                        basename(file), "::\t", appendLF = FALSE,
##                        domain = NA)
##                    ext <- fileExt(file)
##                    if (basename(file) != paste0(name, ".", ext))
##                      found <- FALSE
##                    else {
##                      found <- TRUE
##                      zfile <- file
##                      zipname <- file.path(dirname(file), "Rdata.zip")
##                      if (file.exists(zipname)) {
##                        Rdatadir <- tempfile("Rdata")
##                        dir.create(Rdatadir, showWarnings = FALSE)
##                        topic <- basename(file)
##                        rc <- .External(C_unzip, zipname, topic,
##                          Rdatadir, FALSE, TRUE, FALSE, FALSE)
##                        if (rc == 0L)
##                          zfile <- file.path(Rdatadir, topic)
##                      }
##                      if (zfile != file)
##                        on.exit(unlink(zfile))
##                      switch(ext, R = , r = {
##                        library("utils")
##                        sys.source(zfile, chdir = TRUE, envir = tmp_env)
##                      }, RData = , rdata = , rda = load(zfile,
##                        envir = tmp_env), TXT = , txt = , tab = ,
##                        tab.gz = , tab.bz2 = , tab.xz = , txt.gz = ,
##                        txt.bz2 = , txt.xz = assign(name, my_read_table(zfile,
##                          header = TRUE, as.is = FALSE), envir = tmp_env),
##                        CSV = , csv = , csv.gz = , csv.bz2 = ,
##                        csv.xz = assign(name, my_read_table(zfile,
##                          header = TRUE, sep = ";", as.is = FALSE),
##                          envir = tmp_env), found <- FALSE)
##                    }
##                    if (found)
##                      break
##                 }
##                 if (verbose)
##                   message(if (!found)
##                     "*NOT* ", "found", domain = NA)
##             }
##             if (found)
##                 break
##         }
##         if (!found) {
##             warning(gettextf("data set %s not found", sQuote(name)),
##                 domain = NA)
##         }
##         else if (!overwrite) {
##             for (o in ls(envir = tmp_env, all.names = TRUE)) {
##                 if (exists(o, envir = envir, inherits = FALSE))
##                   warning(gettextf("an object named %s already exists and will not be overwrit
##                     sQuote(o)))
##                 else assign(o, get(o, envir = tmp_env, inherits = FALSE),
##                   envir = envir)
##             }
##             rm(tmp_env)
##         }
##     }
##     invisible(names)
## }
## <bytecode: 0x7fbba7e880a8>
## <environment: namespace:utils>
```

- delivery element: transport power from one place to another
- conversion element: convert power from or to another form

Power grids move electricity through delivery elements to balance conversion elements.

- slack bus
- PQ bus
- PV bus

It is sufficient to model an RDF with one kind of buses and one kind of edges

- attribute associated with bus: voltage, current (power) injection
- attribute associated with edge: current (power) flow

Chapter 3

Two Special Concepts for Power Flow

3.1 Channel

3.2 Snapshot

input: real power injections at all channels of PQ buses output: voltages, current flow, power flow

Zero-load snapshot is the snapshot where power injections at all the channels are zero and voltages equalt o rated voltages in corresponding phases.

Deka et al. (2017)

Bibliography

Deka, D., Backhaus, S., and Chertkov, M. (2017). Structure learning in power distribution networks. *IEEE Transactions on Control of Network Systems*, 5(3):1061–1074.