Appendix C. Creating Datasets for the R-Package 'wrv'

Contents

Figures

Introduction

This vignette explains the processing steps for creating R datasets in the **wrv** package. Datasets are processed at two levels: *level 1* is unprocessed data mapped on a uniform space-time grid scale; and *level 2* results from analyses of level 1 data. Considerable effort was placed on minimizing the number and complexity of processing steps required for dataset creation. However, some datasets (such as the level 2 datasets) are necessary for parameter estimation, and computationally too expensive to recreate during each iteration of model calibration; including these datasets in the **wrv** package avoids these long run times. It is assumed that the reader of this vignette is familiar with the R-programming language and has read help documentation for functions and datasets in the **wrv** package (appendix B).

R Environment

Load the following packages into the current R session:

```
library("rgdal")   # bindings for the geospatial data abstraction library
library("raster")  # gridded spatial data toolkit
```

Set a **raster** package option to prevent the standardization of raster names:

```
rasterOptions(standardnames = FALSE)
```

The memory requirement for running R code in this vignette is about 10 gigabytes.

Input/Output Paths

Package datasets are primarily created from unprocessed data files located on GitHub, a web-based distributed revision control system. Download these files to a temporary directory and uncompress the ZIP files.

```
url <- "https://github.com/USGS-R/wrv.git"
path <-  file.path(tempdir(), basename(tools::file_path_sans_ext(url)))
git2r::clone(url, path, progress = FALSE)
dir.in <- file.path(path, "inst/extdata")
files <- list.files(dir.in, pattern = "*.zip$", full.names = TRUE, recursive = TRUE)
for (i in files) unzip(i, exdir = dirname(i))
```

The data file containing land-surface elevations was deemed too large in file size (about 500 megabytes) to be placed in the package repository. These elevations are part of The National Map (TNM) 1/3-arc-second raster and available in a ArcGRID file format. Download the ZIP file to a temporary directory and uncompress.

```
ftp <- "ftp://rockyftp.cr.usgs.gov/vdelivery/Datasets/Staged/NED/13/ArcGrid/n44w115.zip"
file <- file.path(tempdir(), basename(ftp))
download.file(ftp, file)
unzip(file, exdir = dir.in)
```

Output from this vignette is placed in the current working directory.

```
dir.create(dir.out <- "data", showWarnings = FALSE)
```

Space-Time Grid Scale

The length and time dimensions for datasets are in units of meters and days, respectively. Conversion factors are listed with an explanation of how they are used:

```
in.to.m              <- 0.0254          # inches to meters
ft.to.m              <- 0.3048          # feet to meters
mm.to.m              <- 0.001           # millimeters to meters
mi2.to.m2            <- 2589990         # square miles to square meters
af.to.m3             <- 1233.48185532   # acre-feet to cubic meters
in.per.y.to.m.per.d  <- 6.95429e-05     # inches per year to meters per day
af.per.y.to.m3.per.d <- 3.377           # acre-feet per year to cubic meters per day
cfs.to.m3.per.d      <- 2446.57555      # cubic feet per second to cubic meters per day
```

The common coordinate reference system (CRS) applied to all spatial datasets is the Idaho Transverse Mercator projection (IDTM83). PROJ.4 projection arguments are used to specify a CRS in R. The CRS that all unprocessed data are converted into is specified as:

```
crs <- CRS(paste("+proj=tmerc +lat_0=42 +lon_0=-114 +k=0.9996 +x_0=2500000 +y_0=1200000",
                 "+datum=NAD83 +units=m +no_defs +ellps=GRS80 +towgs84=0,0,0"))
```

The common spatial grid applied to all gridded datasets is composed of 565 rows and 429 columns, and has a constant cell size of 100 meters by 100 meters.

```
ext <- extent(2453200, 2496100, 1344139, 1400639)  # xmin, xmax, ymin, ymax in IDTM
spatial.grid <- raster(crs = crs, ext = ext, resolution = 100)
```

Gridded data are available at a higher resolution (smaller cells) than the resolution of the model grid. Projecting the higher resolution data into the model grid first requires the projection of this data into a high resolution spatial grid of comparable cell size. The high resolution spatial grid is defined using a constant cell size of 20 meters by 20 meters.

```
high.res.spatial.grid <- disaggregate(spatial.grid, fact = 5L)
```

The transient model simulates groundwater flow from 1995 through 2010, using monthly stress periods.

```
tr.interval <- as.Date(c("1995-01-01", "2011-01-01"), tz = "MST")
tr.stress.periods <- seq(tr.interval[1] , tr.interval[2], "1 month")
yr.mo <- format(head(tr.stress.periods, -1), "%Y%m")
yr.mo.irr <- yr.mo[months(head(tr.stress.periods, -1), abbreviate = TRUE) %in%
                   c("Apr", "May", "Jun", "Jul", "Aug", "Sep", "Oct")]
```

Level 1 Data

Datasets processed at level 1 are described as unprocessed data (that is, data read from files or specified in this vignette) mapped on a uniform space-time grid scale. Unprocessed data that is redundant, or deemed unnecessary for model processing or quality assurance, is removed. A few of the level 1 datasets require supplemental processing steps; descriptions of these steps are included alongside the relevant 'code chunks'. Variable names, for the most part, are maintained between unprocessed and processed data.

Tables

Canal seepage (canal.seep)

Canal seepage as a fraction of diversions for irrigation entities in the Wood River Valley (WRV).

```
file <- file.path(dir.in, "canal/canal.seep.csv")
canal.seep <- read.csv(file, strip.white = TRUE)
save(canal.seep, file = file.path(dir.out, "canal.seep.rda"), compress = "xz")
```

Combined surface-water irrigation diversions (comb.sw.irr)

Supplemental groundwater rights and associated surface-water rights.

```
file <- file.path(dir.in, "div/comb.sw.irr.csv")
comb.sw.irr <- read.csv(file, strip.white = TRUE)
comb.sw.irr$Pdate <- as.Date(comb.sw.irr$Pdate, format = "%m/%d/%Y")
comb.sw.irr$MaxDivRate <- comb.sw.irr$MaxDivRate * cfs.to.m3.per.d
save(comb.sw.irr, file = file.path(dir.out, "comb.sw.irr.rda"), compress = "xz")
```

Evapotranspiration methods (et.method)

Methods used to calculate monthly distributions of evapotranspiration rate.

```
file <- file.path(dir.in, "et/et.method.csv")
et.method <- read.csv(file, strip.white = TRUE)
et.method$YearMonth <- as.character(et.method$YearMonth)
save(et.method, file = file.path(dir.out, "et.method.rda"), compress = "xz")
```

Groundwater diversions (div.gw)

Groundwater diversions recorded by Water District 37 or municipal water providers. Groundwater is diverted from the aquifer by means of either pumping wells or flowing artesian wells.

```
file <- file.path(dir.in, "div/div.gw.csv")
div.gw <- read.csv(file, strip.white = TRUE)
div.gw$YearMonth <- as.factor(div.gw$YearMonth)
div.gw$GWDiv <- div.gw$GWDiv_af * af.to.m3
div.gw$GWDiv_af <- NULL
div.gw[is.na(div.gw$GWDiv), "GWDiv"] <- 0
save(div.gw, file = file.path(dir.out, "div.gw.rda"), compress = "xz")
```

Irrigation efficiency (efficiency)

Irrigation efficiency for irrigation entities.

```
file <- file.path(dir.in, "irr/efficiency.csv")
efficiency <- read.csv(file, strip.white = TRUE)
save(efficiency, file = file.path(dir.out, "efficiency.rda"), compress = "xz")
```

Irrigation lands for a given year (irr.lands.year)

The annual land classification for irrigation practices is only available for select years. For missing years, this dataset provides substitute years when land-classification was available.

```
file <- file.path(dir.in, "irr/irr.lands.year.csv")
irr.lands.year <- read.csv(file, strip.white = TRUE, colClasses = "character")
save(irr.lands.year, file = file.path(dir.out, "irr.lands.year.rda"), compress = "xz")
```

Snow Water Equivalent (swe)

Average daily snow water equivalent (SWE) at weather stations in the WRV and surrounding areas.

```
file <- file.path(dir.in, "precip/swe.choco.csv")
swe.choco <- read.csv(file, strip.white = TRUE)
swe.choco$MonthDay <- format(as.Date(swe.choco$Date, "%m/%d/%Y"), "%m%d")
swe.choco$SWE <- swe.choco$SWE_in * in.to.m
file <- file.path(dir.in, "precip/swe.hailey.csv")
swe.hailey <- read.csv(file, strip.white = TRUE)
swe.hailey$MonthDay <- format(as.Date(swe.hailey$Date, "%m/%d/%Y"), "%m%d")
swe.hailey$SWE <- swe.hailey$SWE_in * in.to.m
file <- file.path(dir.in, "precip/swe.picabo.csv")
swe.picabo <- read.csv(file, strip.white = TRUE)
swe.picabo$MonthDay <- format(as.Date(swe.picabo$Date, "%m/%d/%Y"), "%m%d")
swe.picabo$SWE <- swe.picabo$SWE_in * in.to.m
```

Aggregate SWE data by day in a year:

```
swe.choco  <- aggregate(swe.choco$SWE, list(swe.choco$MonthDay), mean)
swe.hailey <- aggregate(swe.hailey$SWE, list(swe.hailey$MonthDay), mean)
swe.picabo <- aggregate(swe.picabo$SWE, list(swe.picabo$MonthDay), mean)
```

Combine datasets and write to disk:

```
swe <- swe.choco[order(swe.choco[[1]]), ]
swe <- dplyr::left_join(swe, swe.hailey, by = "Group.1")
swe <- dplyr::left_join(swe, swe.picabo, by = "Group.1")
names(swe) <- c("MonthDay", "Choco", "Hailey", "Picabo")
save(swe, file = file.path(dir.out, "swe.rda"), compress = "xz")
```

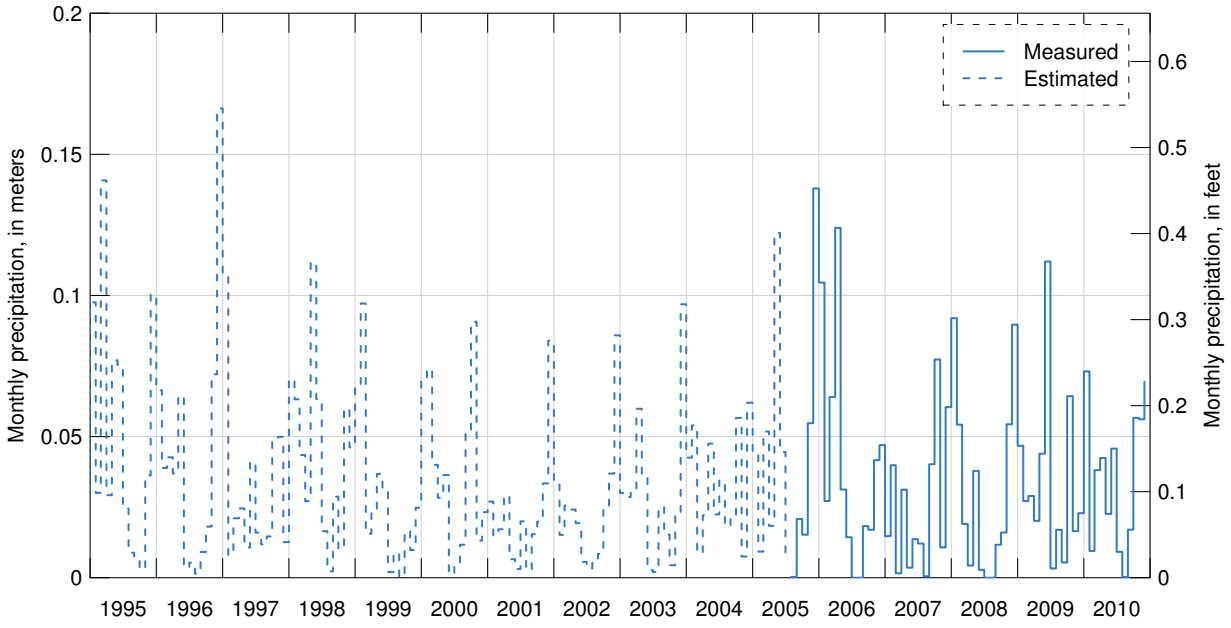

Figure C1. Monthly precipitation depth at the Hailey HADS weather station in the Wood River Valley aquifer system, south-central Idaho.

Precipitation Rate (precipitation)

Precipitation rates in the WRV and surrounding areas. Combine precipitation records into a single data table:

```
file <- file.path(dir.in, "precip/precip.csv")
d <- read.csv(file, strip.white = TRUE)
d$Ketchum <- d$Ketchum_ft * ft.to.m
d$Hailey  <- d$Hailey_ft  * ft.to.m
d$Picabo  <- d$Picabo_ft  * ft.to.m
d$Ketchum_ft <- NULL
d$Hailey_ft  <- NULL
d$Picabo_ft  <- NULL
```

Create a linear regression model between the average of the precipitation depth recorded at the Picabo and Ketchum weather stations, and the precipitation depth recorded at the Hailey weather station:

```
x <- apply(d[, c("Picabo", "Ketchum")], 1, mean)
y <- d$Hailey
LM <- lm(y ~ x)
```

A strong positive correlation (R-squared of 0.934) indicates that the regression model may be used to estimate missing data at the Hailey weather station (fig. C1).

```
is.na.hailey <- is.na(y)
d$Hailey[is.na.hailey] <- predict(LM, data.frame(x))[is.na.hailey]
precipitation <- d
```

Use a monthly precipitation redistribution model to account for frozen precipitation (snow):

```
d <- precipitation
mo <- month.abb[as.integer(substr(precipitation$YearMonth, 5, 6))]
for (i in seq_along(mo)) {
  if (mo[i] == "Nov") {
    precipitation$Ketchum[i] <- d$Ketchum[i] * 0.25
    precipitation$Hailey[i]  <- d$Hailey[i]  * 0.75
    precipitation$Picabo[i]  <- d$Picabo[i]  * 0.75
  } else if (mo[i] %in% c("Dec", "Jan")) {
    precipitation$Ketchum[i] <- d$Ketchum[i] * 0.25
    precipitation$Hailey[i]  <- d$Hailey[i]  * 0.25
    precipitation$Picabo[i]  <- d$Picabo[i]  * 0.25
  } else if (mo[i] == "Feb") {
    precipitation$Ketchum[i] <- d$Ketchum[i] * 0.25
    precipitation$Hailey[i]  <- d$Hailey[i]  * 0.50
    precipitation$Picabo[i]  <- d$Picabo[i]  * 0.75
  } else if (mo[i] == "Mar") {
    precipitation$Ketchum[i] <- d$Ketchum[i] * 0.25
    precipitation$Hailey[i]  <- sum(d$Hailey[(i - 4L):i] * c(0.25, 0.75, 0.75, 0.50, 1))
    precipitation$Picabo[i]  <- sum(d$Picabo[(i - 4L):i] * c(0.25, 0.75, 0.75, 0.25, 1))
  } else if (mo[i] == "Apr") {
    precipitation$Ketchum[i] <- sum(d$Ketchum[(i - 5L):i] * c(rep(0.75, 5), 1))
  }
}
precipitation <- dplyr::left_join(d, precipitation, by = "YearMonth")
sites <- c("Ketchum", "Hailey", "Picabo")
names(precipitation) <- c("YearMonth", paste0(sites, ".raw"), sites)
```

Remove precipitation records that occurred outside the model simulation period:

```
date.time <- as.Date(paste0(precipitation[, "YearMonth"], "01"), "%Y%m%d")
precipitation <- precipitation[date.time >= tr.interval[1] & date.time < tr.interval[2], ]
```

Add precipitation zone meta data:

```
d <- data.frame(YearMonth = as.factor(rep(as.character(precipitation$YearMonth), 3)),
                PrecipZone = rep(sites, each = nrow(precipitation)),
                Precip = NA, Precip.raw = NA)
d[d$PrecipZone == "Ketchum", 3:4] <- precipitation[, c("Ketchum", "Ketchum.raw")]
d[d$PrecipZone == "Hailey",  3:4] <- precipitation[, c("Hailey", "Hailey.raw")]
d[d$PrecipZone == "Picabo",  3:4] <- precipitation[, c("Picabo", "Picabo.raw")]
```

Save the dataset to disk:

```
precipitation <- d
save(precipitation, file = file.path(dir.out, "precipitation.rda"), compress = "xz")
```

Priority cuts (priority.cuts)

Priority cut dates applied to Big Wood River above Magic Reservoir and Silver Creek by Water District 37 and 37M at the end of each month.

```
file <- file.path(dir.in, "div/priority.cuts.csv")
priority.cuts <- read.csv(file, strip.white = TRUE)
priority.cuts$YearMonth <- as.factor(priority.cuts$YearMonth)
priority.cuts$Pdate_BWR <- as.Date(priority.cuts$Pdate_BWR, format = "%m/%d/%Y")
priority.cuts$Pdate_SC <- as.Date(priority.cuts$Pdate_SC, format = "%m/%d/%Y")
save(priority.cuts, file = file.path(dir.out, "priority.cuts.rda"), compress = "xz")
```

Surface-water diversions (div.sw)

Surface-water diversions recorded by Water District 37 or municipal water providers.

```
file <- file.path(dir.in, "div/div.sw.csv")
div.sw <- read.csv(file, strip.white = TRUE)
div.sw$YearMonth <- as.factor(div.sw$YearMonth)
div.sw$SWDiv <- div.sw$SWDiv_af * af.to.m3
div.sw$SWDiv_af <- NULL
div.sw[is.na(div.sw$SWDiv), "SWDiv"] <- 0
save(div.sw, file = file.path(dir.out, "div.sw.rda"), compress = "xz")
```

Wastewater treatment plant diversions (div.ww)

Discharge from wastewater treatment plants.

```
file <- file.path(dir.in, "div/div.ww.csv")
div.ww <- read.csv(file, strip.white = TRUE)
div.ww$YearMonth <- as.factor(div.ww$YearMonth)
div.ww$WWDiv <- div.ww$WWTP_af * af.to.m3
div.ww$WWTP_af <- NULL
div.ww[is.na(div.ww$WWDiv), "WWDiv"] <- 0
save(div.ww, file = file.path(dir.out, "div.ww.rda"), compress = "xz")
```

Daily mean discharge at streamgages (gage.disch)

Daily mean discharge at streamgages in the WRV: Big Wood River near Ketchum, Idaho (13135500); Big Wood River at Hailey, Idaho (13139510); and Big Wood River at Stanton Crossing near Bellevue, Idaho (13140800).

```
FUN <- function(i) {
  file <- file.path(dir.in, "gage", i)
  d <- read.csv(file, colClasses = "character", strip.white = TRUE)
  d$Date <- as.Date(d$Date, format = "%Y-%m-%d")
  d$Disch <- suppressWarnings(as.numeric(d$Disch_cfs)) * cfs.to.m3.per.d
  return(d[substr(d$Code, 1, 1) == "A" & !is.na(d$Disch), c("Date", "Disch")])
}
gage.13135500.disch <- FUN("gage.13135500.disch.csv")
gage.13139510.disch <- FUN("gage.13139510.disch.csv")
gage.13140800.disch <- FUN("gage.13140800.disch.csv")
```

Combine discharge records into a single data table:

```
dlim <- range(c(gage.13135500.disch$Date, gage.13139510.disch$Date,
                gage.13140800.disch$Date))
d <- data.frame(Date=seq(dlim[1], dlim[2], by = "day"))
d <- dplyr::left_join(d, gage.13135500.disch, by = "Date")
d <- dplyr::left_join(d, gage.13139510.disch, by = "Date")
d <- dplyr::left_join(d, gage.13140800.disch, by = "Date")
colnames(d) <- c("Date", "13135500", "13139510", "13140800")
gage.disch <- d
save(gage.disch, file = file.path(dir.out, "gage.disch.rda"), compress = "xz")
```

Daily mean gage height at streamgages (gage.height)

Daily mean gage height at streamgages in the WRV: Big Wood River near Ketchum, Idaho (13135500); Big Wood River at Hailey, Idaho (13139510); and Big Wood River at Stanton Crossing near Bellevue, Idaho (13140800) (fig. C2).

```
FUN <- function(i) {
  file <- file.path(dir.in, "gage", i)
  d <- read.csv(file, colClasses = "character", strip.white = TRUE)
  d$DateTime <- strptime(d$DateTime, "%Y-%m-%d %H:%M", tz = "MST")
  d$Date <- as.Date(d$DateTime)
  d$Height <- suppressWarnings(as.numeric(d$Height_ft)) * ft.to.m
  d <- d[substr(d$Code, 1, 1) %in% c("W", "R", "A") & !is.na(d$Height), ]
  d <- aggregate(d$Height, list(d$Date), mean, na.rm = TRUE)
  names(d) <- c("Date", "Height")
  return(d)
}
gage.13135500.height <- FUN("gage.13135500.height.csv")
gage.13139510.height <- FUN("gage.13139510.height.csv")
gage.13140800.height <- FUN("gage.13140800.height.csv")
```

Combine gage-height records into a single data table:

```
dlim <- range(c(gage.13135500.height$Date, gage.13139510.height$Date,
                gage.13140800.height$Date))
d <- data.frame(Date=seq(dlim[1], dlim[2], by = "day"))
d <- dplyr::left_join(d, gage.13135500.height, by = "Date")
d <- dplyr::left_join(d, gage.13139510.height, by = "Date")
d <- dplyr::left_join(d, gage.13140800.height, by = "Date")
colnames(d) <- c("Date", "13135500", "13139510", "13140800")
```

Remove negative values of gage height (n = 2):

```
d[d < 0] <- NA
```

Create a linear regression model between gage-height data recorded at the Hailey gage and Near Ketchum gage:

```
x <- d[["13139510"]]
y <- d[["13135500"]]
LM <- lm(y ~ x)
```

A strong positive correlation (R-squared of 0.964) indicates that the regression model may be used to estimate missing data at the Near Ketchum gage.

```
is.na.13135500 <- is.na(y)
d[["13135500"]][is.na.13135500] <- predict(LM, data.frame(x))[is.na.13135500]
```

Missing data at the Stanton Crossing near Bellevue gage are replaced with average gage-height values recorded at this gage. Substantial seepage losses and surface-water diversions between the Hailey gage and Stanton Crossing near Bellevue gage make regression inappropriate.

```
jday <- as.integer(julian(as.Date(paste0("1900-", format(d$Date, "%m-%d"))),
                          origin = as.Date("1899-12-31")))
m <- cbind(jday, height = d[["13140800"]])
m <- m[rowSums(is.na(m)) == 0, ]
m0 <- m[m[, "jday"] > 300, ]
m1 <- m[m[, "jday"] < 66, ]
m0[, "jday"] <- m0[, "jday"] - 365
m1[, "jday"] <- m1[, "jday"] + 365
LPM <- loess(height ~ jday, data.frame(rbind(m0, m, m1)), span = 1 / 35)
ave.heights <- predict(LPM, newdata = 1:365)[jday]
is.na.13140800 <- is.na(d[["13140800"]])
d[["13140800"]][is.na.13140800] <- ave.heights[is.na.13140800]
```

Save the dataset to disk:

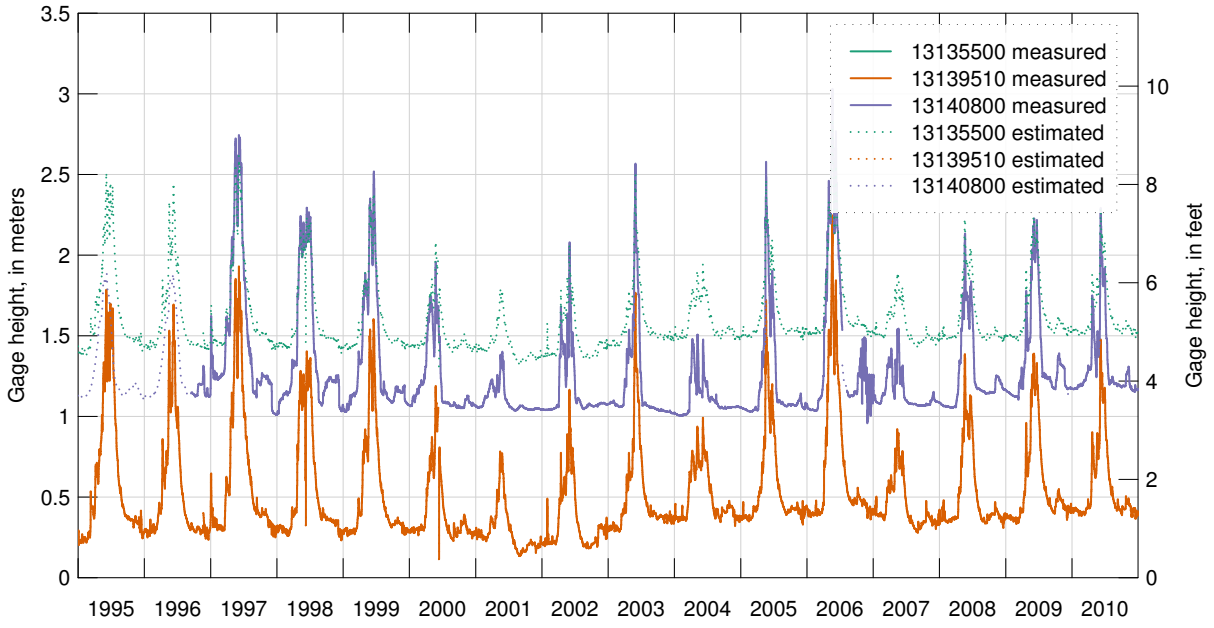

Figure C2. Gage heights recorded at streamgages along the Big Wood River.

```
gage.height <- d
save(gage.height, file = file.path(dir.out, "gage.height.rda"), compress = "xz")
```

Points of diversion for groundwater (pod.gw)

Points of diversion for groundwater.

```
file <- file.path(dir.in, "div/pod.gw.csv")
d <- read.csv(file, strip.white = TRUE, stringsAsFactors = FALSE)
d$Pdate <- as.Date(d$PriorityDa, format = "%m/%d/%Y")
d$IrrRate <- d$IRRcfs * cfs.to.m3.per.d
columns <- c("WMISNumber", "WaterRight", "EntityName", "EntitySrce", "Pdate", "IrrRate")
pod.gw <- d[, columns]
save(pod.gw, file = file.path(dir.out, "pod.gw.rda"), compress = "xz")
```

Recharge at miscellaneous seepage sites (misc.seepage)

Recharge from miscellaneous seepage sites.

```
file <- file.path(dir.in, "misc.seepage.csv")
d <- read.csv(file, strip.white = TRUE)
d$Rech <- d$Rech_af * af.to.m3
FUN <- function(i) {
  x <- d[d$YearMonth == i, c("RechSite", "Rech")]
  colnames(x) <- c("RechSite", i)
  return(x)
}
l <- lapply(yr.mo, FUN)
misc.seepage <- Reduce(function(x, y) merge(x, y, all = TRUE, by = "RechSite"), l)
save(misc.seepage, file = file.path(dir.out, "misc.seepage.rda"), compress = "xz")
```

Hydraulic properties of hydrogeologic zones (zone.properties)

Hydraulic properties for each hydrogeologic zone.

```
file <- file.path(dir.in, "zone.properties.csv")
d <- read.csv(file, strip.white = TRUE, stringsAsFactors = FALSE)
d$hk <- d$hk_ft.per.d * ft.to.m
d$hk_ft.per.d <- NULL
d$ss <- d$ss_per.ft / ft.to.m
d$ss_per.ft <- NULL
zone.properties <- d
save(zone.properties, file = file.path(dir.out, "zone.properties.rda"), compress = "xz")
```

Dry river bed and stream fed creek conditions (drybed)

Spring fed creek conditions are specified for select stream reaches. The mathematical representation of spring fed creeks using the MODFLOW river package is identical to the mathematical representation for a dry stream bed.

```
file <- file.path(dir.in, "perennial.reaches.csv")
perennial.reaches <- read.csv(file, colClasses = "character", strip.white = TRUE)[, 1]
drybed <- as.data.frame(matrix(NA, nrow = length(perennial.reaches), ncol = length(yr.mo)))
colnames(drybed) <- yr.mo
rownames(drybed) <- perennial.reaches
drybed[perennial.reaches, ] <- TRUE
```

Stream reaches on the Big Wood River between Glendale and Wood River Ranch are episodically dry; dry bed conditions are specified each year beginning in the first month when the entire flow of the Big Wood River is diverted into the Bypass Canal before the 16th of the month and ending at the end of October. Between Wood River Ranch and Stanton Crossing, the Big Wood River gains water from springs and seeps, thus this reach acts as a spring fed creek when dry bed conditions are specified between Glendale and Wood River Ranch and there is no return flow from the Bypass Canal to the Big Wood River.

```
file <- file.path(dir.in, "canal/bypass.canal.op.csv")
d <- read.csv(file, colClasses = "character", strip.white = TRUE)
date1 <- as.Date(d$StartDate, tz = "MST")
date2 <- as.Date(d$EndDate, tz = "MST")
FUN <- function(i) {
  d <- as.integer(format(i, format = "%d"))
  m <- format(i, format = "%m")
  while (format(i, format = "%m") == m) i <- i + 1L
  return(d / as.integer(format(i - 1L, format = "%d")))
}
frac1 <- vapply(date1, FUN, 0)
frac2 <- vapply(date2, FUN, 0)
date1[frac1 > 0.5] <- date1[frac1 > 0.5] + 16L
date2[frac2 < 0.5] <- date2[frac2 < 0.5] - 16L
date1 <- as.Date(paste0(format(date1, format = "%Y-%m"), "-01"))
date2 <- as.Date(paste0(format(date2, format = "%Y-%m"), "-01"))
FUN <- function(i) format(seq(date1[i], date2[i], by = "month"), format = "%Y%m")
is.drybed <- yr.mo %in% unlist(lapply(seq_along(date1), FUN))
episodic.reaches <- c("Big Wood, Glendale to Sluder",
                      "Big Wood, Sluder to Wood River Ranch",
                      "Big Wood, Wood River Ranch to Stanton Crossing")
for (i in episodic.reaches) drybed[i, ] <- is.drybed
```

During the month of October the water district stops monitoring diversions and much of the water diverted into the Bypass Canal is returned to the Big Wood River at Wood River Ranch. Therefore, flows in the Big Wood, Wood River Ranch to Stanton Crossing reach are accounted for in the model.

```
drybed["Big Wood, Wood River Ranch to Stanton Crossing",
       substr(colnames(drybed), 5, 6) == "10"] <- FALSE
```

Save the dataset to disk:

```
drybed <- data.frame(Reach = rownames(drybed), drybed, check.names = FALSE,
                     row.names = NULL, stringsAsFactors = FALSE)
save(drybed, file = file.path(dir.out, "drybed.rda"), compress = "xz")
```

Groundwater-level measurements (obs.wells.head)

Groundwater-level measurements recorded in observation wells in the WRV. Values used as observations in parameter estimation.

```
file <- file.path(dir.in, "opt/obs.wells.head.csv")
d <- read.csv(file, strip.white = TRUE, stringsAsFactors = FALSE)
d$DateTime <- as.POSIXct(d$DateTime, tz = "MST", format = "%Y-%m-%d %H:%M:%S")
d$Head <- as.numeric(d$Head_m)
d$Head_m <- NULL
obs.wells.head <- d
save(obs.wells.head, file = file.path(dir.out, "obs.wells.head.rda"), compress = "xz")
```

Stream-aquifer flow exchange along river reaches (reach.recharge)

Stream-aquifer flow exchange along river reaches specified as aquifer recharge. Values used as observations in parameter estimation.

```
file <- file.path(dir.in, "opt/reach.recharge.csv")
d <- read.csv(file, strip.white = TRUE, stringsAsFactors = FALSE)
d$YearMonth <- as.character(d$YearMonth)
d$nKet_Hai <- d$nKet_Hai_cfs * cfs.to.m3.per.d
d$nKet_Hai_cfs <- NULL
d$Hai_StC <- d$Hai_StC_cfs * cfs.to.m3.per.d
d$Hai_StC_cfs <- NULL
d$WillowCr <- d$WillowCr_cfs * cfs.to.m3.per.d
d$WillowCr_cfs <- NULL
d$SilverAbv <- d$SilverAbv_cfs * cfs.to.m3.per.d
d$SilverAbv_cfs <- NULL
d$SilverBlw <- d$SilverBlw_cfs * cfs.to.m3.per.d
d$SilverBlw_cfs <- NULL
reach.recharge <- d
save(reach.recharge, file = file.path(dir.out, "reach.recharge.rda"), compress = "xz")
```

Stream-aquifer flow exchange along river subreaches (subreach.recharge)

Stream-aquifer flow exchange along river subreaches specified as aquifer recharge. Values used as observations in parameter estimation.

```
file <- file.path(dir.in, "opt/subreach.recharge.csv")
d <- read.csv(file, strip.white = TRUE, stringsAsFactors = FALSE)
d[, c("Aug", "Oct", "Mar")] <- d[, c("Aug_cfs", "Oct_cfs", "Mar_cfs")] * cfs.to.m3.per.d
d$Aug_cfs <- NULL
d$Oct_cfs <- NULL
d$Mar_cfs <- NULL
subreach.recharge <- d
save(subreach.recharge, file = file.path(dir.out, "subreach.recharge.rda"),
     compress = "xz")
```

PEST sensitivity analysis (sensitivity)

Calibrated parameter values and composite sensitivities generated by PEST during its last iteration.

```
file <- file.path(dir.in, "opt/sensitivity.csv")
sensitivity <- read.csv(file, strip.white = TRUE)
sensitivity$parameter.name <- as.character(sensitivity$parameter.name)
rel <- with(sensitivity, comp.sens * abs(value))  # Realtive Composite Sensitivity
is.log <- sensitivity$parameter.desc %in% c("Horizontal hydraulic conductivity",
                                            "Storage coefficient",
                                            "Riverbed conductance",
                                            "Drain conductance")
rel[is.log] <- with(sensitivity, comp.sens * abs(log10(value)))[is.log]
sensitivity$rel.comp.sens <- rel
save(sensitivity, file = file.path(dir.out, "sensitivity.rda"), compress = "xz")
```

Points

Cities and towns (cities)

Cities and towns in the WRV and surrounding areas.

```
path <- file.path(dir.in, "decorative")
cities <- readOGR(path, "cities", verbose = FALSE, integer64 = "allow.loss")
cities <- spTransform(cities, crs)
cities <- cities[cities@data$FEATURE_NA != "Elkhorn Village", ]
save(cities, file = file.path(dir.out, "cities.rda"), compress = "xz")
```

Map labels (map.labels)

Map labels in the WRV and surrounding areas.

```
file <- file.path(dir.in, "decorative/map.labels.csv")
map.labels <- read.csv(file, strip.white = TRUE, stringsAsFactors = FALSE)
map.labels$label <- sub("\\\\n", "\\\n", map.labels$label)
coordinates(map.labels) <- 1:2
colnames(map.labels@coords) <- c("x", "y")
proj4string(map.labels) <- CRS("+init=epsg:4326")
map.labels <- spTransform(map.labels, crs)
save(map.labels, file = file.path(dir.out, "map.labels.rda"), compress = "xz")
```

Miscellaneous Locations (misc.locations)

Miscellaneous locations in the Bellevue triangle area.

```
file <- file.path(dir.in, "decorative/misc.locations.csv")
misc.locations <- read.csv(file, strip.white = TRUE, stringsAsFactors = FALSE)
misc.locations$label <- sub("\\\\n", "\\\n", misc.locations$label)
coordinates(misc.locations) <- 1:2
colnames(misc.locations@coords) <- c("x", "y")
proj4string(misc.locations) <- CRS("+init=epsg:4326")
misc.locations <- spTransform(misc.locations, crs)
save(misc.locations, file = file.path(dir.out, "misc.locations.rda"), compress = "xz")
```

Weather Stations (weather.stations)

Weather stations in the WRV and surrounding areas.

```
file <- file.path(dir.in, "precip/weather.stations.csv")
weather.stations <- read.csv(file, strip.white = TRUE, stringsAsFactors = FALSE)
weather.stations$elevation <- weather.stations$elevation_ft * ft.to.m
weather.stations$elevation_ft <- NULL
weather.stations$url <- NULL
coordinates(weather.stations) <- 1:2
colnames(weather.stations@coords) <- c("x", "y")
proj4string(weather.stations) <- CRS("+init=epsg:4326")
weather.stations <- spTransform(weather.stations, crs)
save(weather.stations, file = file.path(dir.out, "weather.stations.rda"), compress = "xz")
```

Well completions (pod.wells)

Well completions for pumping wells in the WRV aquifer system.

```
path <- file.path(dir.in, "div")
pod.wells <- readOGR(path, "pod.wells", verbose = FALSE, integer64 = "allow.loss")
pod.wells <- spTransform(pod.wells, crs)
d <- pod.wells@data
columns <- c("TopOpen1", "BotOpen1", "TopOpen2", "BotOpen2")
d[, columns] <- d[, columns] * ft.to.m
d[d$TopOpen1 == 0 | d$BotOpen1 == 0, c("TopOpen1", "BotOpen1")] <- NA
d[d$TopOpen2 == 0 | d$BotOpen2 == 0, c("TopOpen2", "BotOpen2")] <- NA
```

A missing well completion is assumed identical to the completion of its nearest-neighbor well.

```
is.pred <- is.na(d$TopOpen1)
dists <- as.matrix(dist(coordinates(pod.wells)))
dists <- dists[!is.pred & d$WellUse %in% "Irrigation", ]
nearest.well <- as.integer(apply(dists, 2, function(i) names(which.min(i))))
d$TopOpen1[is.pred] <- d$TopOpen1[nearest.well[is.pred]]
d$BotOpen1[is.pred] <- d$BotOpen1[nearest.well[is.pred]]
columns <- c("WMISNumber", "WellUse", "TopOpen1", "BotOpen1", "TopOpen2", "BotOpen2")
pod.wells@data <- d[, columns]
save(pod.wells, file = file.path(dir.out, "pod.wells.rda"), compress = "xz")
```

There are a total of 1,243 pumping wells; 1,044 of these wells (83 percent) are missing well completions (fig. C3) and are mostly located in the upper WRV.

Streamgages (streamgages)

Select streamgages in the WRV.

```
path <- file.path(dir.in, "gage")
streamgages <- readOGR(path, "streamgages", verbose = FALSE)
streamgages <- spTransform(streamgages, crs)
save(streamgages, file = file.path(dir.out, "streamgages.rda"), compress = "xz")
```

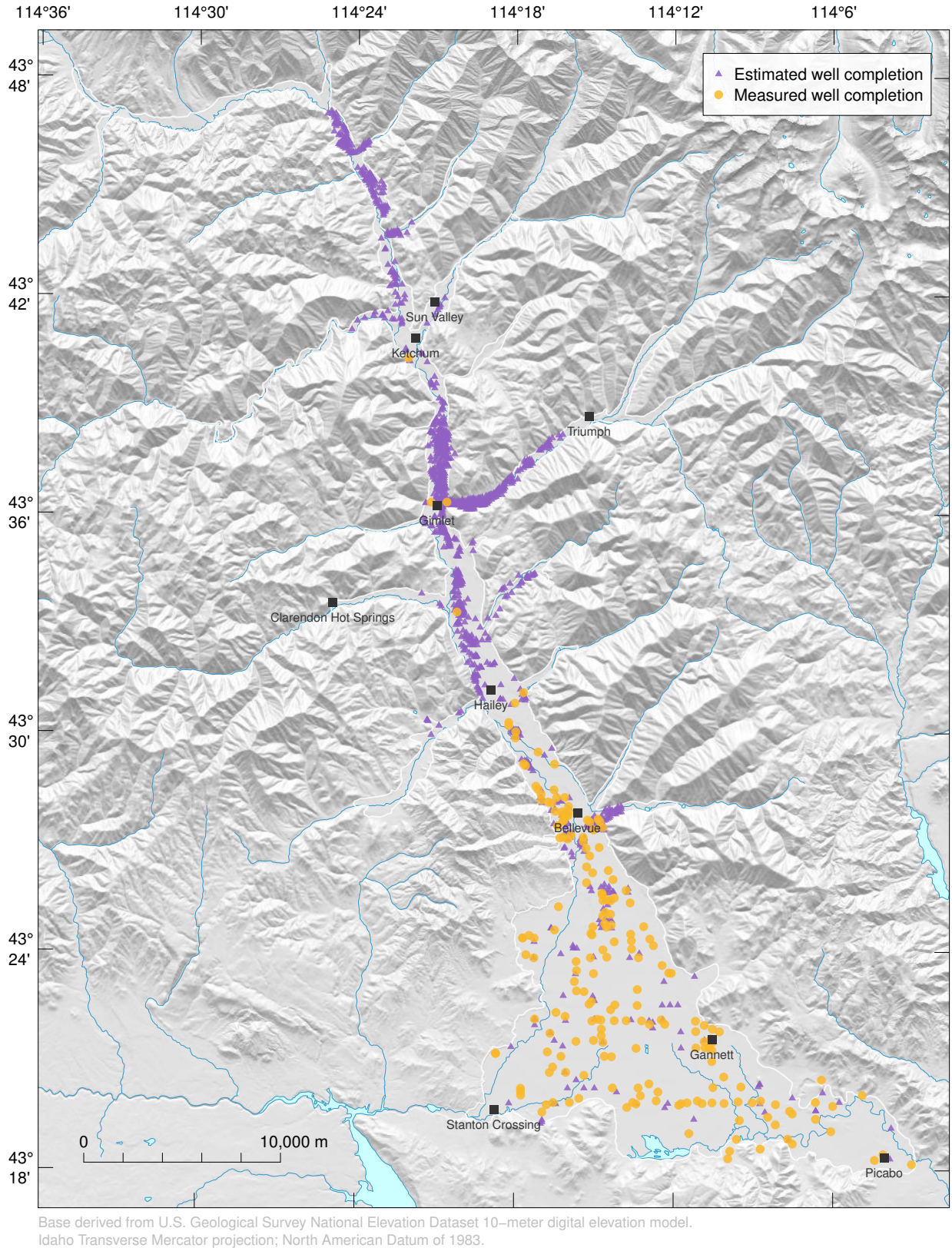

Figure C3. Location of pumping wells and their completion status in the Wood River Valley aquifer system, south-central Idaho.

Observation wells (obs.wells)

Observation wells in the WRV.

```
path <- file.path(dir.in, "opt")
obs.wells <- readOGR(path, "obs.wells", verbose = FALSE, stringsAsFactors = FALSE,
                     integer64 = "allow.loss")
obs.wells <- spTransform(obs.wells, crs)
d <- obs.wells@data
d$COMPLETION <- as.Date(d$COMPLETION, tz = "MST", format = "%Y-%m-%d")
d$ALTMETHOD  <- as.factor(d$ALTMETHOD)
d$XYMETHOD   <- as.factor(d$XYMETHOD)
d$COUNTYNAME <- as.factor(d$COUNTYNAME)
d$desc       <- as.factor(d$desc)
d$TopOpen1 <- as.numeric(d$OPENINGMIN) * ft.to.m
d$BotOpen1 <- as.numeric(d$OPENINGMAX) * ft.to.m
d$TopOpen2 <- NA
d$BotOpen2 <- NA
```

A missing well completion is assumed identical to the completion of its nearest-neighbor well.

```
is.pred <- is.na(d$TopOpen1)
dists <- as.matrix(dist(coordinates(obs.wells)))[!is.pred, ]
nearest.well <- as.integer(apply(dists, 2, function(i) names(which.min(i))))
d$TopOpen1[is.pred] <- d$TopOpen1[nearest.well[is.pred]]
d$BotOpen1[is.pred] <- d$BotOpen1[nearest.well[is.pred]]
obs.wells@data <- d
```

There are a total of 776 observation wells; 2 of these wells (or 0 percent) are missing well completions. Well completion depths are not relevant in the northern portion of the model area where only model layer 1 is active. Well completion data were not compiled for this area.

```
save(obs.wells, file = file.path(dir.out, "obs.wells.rda"), compress = "xz")
```

Stream seepage study (seepage.study)

A Wood River Valley stream seepage study with streamflow measurements made during the months of August 2012, October 2012, and March 2013.

```
path <- file.path(dir.in, "opt")
seepage.study <- readOGR(path, "seepage.study", verbose = FALSE, integer64 = "allow.loss")
seepage.study <- spTransform(seepage.study, crs)
d <- seepage.study@data
d$Aug <- d$Aug_cfs * cfs.to.m3.per.d
d$Oct <- d$Oct_cfs * cfs.to.m3.per.d
d$Mar <- d$Mar_cfs * cfs.to.m3.per.d
d$Aug_cfs <- NULL
d$Oct_cfs <- NULL
d$Mar_cfs <- NULL
seepage.study@data <- d
save(seepage.study, file = file.path(dir.out, "seepage.study.rda"), compress = "xz")
```

Diversions, returns, and exchange wells (div.ret.exch)

Location of streamflow diversions, irrigation canal or pond returns, and exchange well returns.

```
path <- file.path(dir.in, "opt")
div.ret.exch <- readOGR(path, "div.ret.exch", verbose = FALSE)
div.ret.exch <- spTransform(div.ret.exch, crs)
save(div.ret.exch, file = file.path(dir.out, "div.ret.exch.rda"), compress = "xz")
```

Pilot points (pilot.points)

Location of pilot points in the model domain.

```
path <- file.path(dir.in, "opt")
pilot.points <- readOGR(path, "pilot.points", verbose = FALSE, integer64 = "allow.loss")
pilot.points <- spTransform(pilot.points, crs)
save(pilot.points, file = file.path(dir.out, "pilot.points.rda"), compress = "xz")
```

Lines

Canals (canals)

Canal systems in the WRV and surrounding areas.

```
path <- file.path(dir.in, "canal")
canals <- readOGR(path, "canals", verbose = FALSE)
canals <- spTransform(canals, crs)
canals@data$Name <- as.character(canals@data$NAME)
canals@data <- canals@data[, c("EntityName", "Name")]
save(canals, file = file.path(dir.out, "canals.rda"), compress = "xz")
```

Stream reaches of the Big Wood River and Silver Creek (river.reaches)

Stream reaches of the Big Wood River and Silver Creek.

```
river.reaches <- readOGR(dir.in, "river.reaches", verbose = FALSE, integer64 = "allow.loss")
river.reaches <- spTransform(river.reaches, crs)
d <- river.reaches@data
d$RchAvg <- d$RchAvg_cfs * cfs.to.m3.per.d
d$BigRAv <- d$BigRAv_cfs * cfs.to.m3.per.d
d$Depth  <- d$Depth_ft  * ft.to.m
d$BedThk <- d$BedThk_ft * ft.to.m
columns <- c("Reach", "BigReach", "DrainRiver", "RchAvg", "BigRAv", "ReachNo",
             "Depth", "BedThk")
river.reaches@data <- d[, columns]
save(river.reaches, file = file.path(dir.out, "river.reaches.rda"), compress = "xz")
```

Streams and rivers (streams.rivers)

Streams and rivers of the WRV and surrounding areas.

```
path <- file.path(dir.in, "decorative")
streams.rivers <- readOGR(path, "rivers", verbose = FALSE, integer64 = "allow.loss")
streams.rivers <- spTransform(streams.rivers, crs)
save(streams.rivers, file = file.path(dir.out, "streams.rivers.rda"), compress = "xz")
```

Tributary streams (tributary.streams)

Tributary streams of the upper WRV and surrounding areas.

```
path <- file.path(dir.in, "decorative")
tributary.streams <- readOGR(path, "tributary.streams", verbose = FALSE,
                             integer64 = "allow.loss")
tributary.streams <- spTransform(tributary.streams, crs)
save(tributary.streams, file = file.path(dir.out, "tributary.streams.rda"),
     compress = "xz")
```

Bypass canal (bypass.canal)

The location of the Bypass Canal.

```
path <- file.path(dir.in, "canal")
bypass.canal <- readOGR(path, "bypass.canal", verbose = FALSE)
bypass.canal <- spTransform(bypass.canal, crs)
bypass.canal <- as(bypass.canal, "SpatialLines")
save(bypass.canal, file = file.path(dir.out, "bypass.canal.rda"), compress = "xz")
```

Groundwater-level contours for October 2006 (wl.200610)

Groundwater-level contours for the unconfined aquifer in the Wood River Valley, south-central Idaho, representing conditions during October 2006.

```
wl.200610 <- readOGR(dir.in, "wl.200610", verbose = FALSE)
wl.200610 <- spTransform(wl.200610, crs)
wl.200610@data$CONTOUR <- wl.200610@data$CONTOUR * ft.to.m
save(wl.200610, file = file.path(dir.out, "wl.200610.rda"), compress = "xz")
```

Major roads (major.roads)

Major roads in the Wood River Valley and surrounding areas.

```
path <- file.path(dir.in, "decorative")
major.roads <- readOGR(path, "major.roads", verbose = FALSE)
major.roads <- spTransform(major.roads, crs)
save(major.roads, file = file.path(dir.out, "major.roads.rda"), compress = "xz")
```

Polygons

Extent of alluvium unit (alluvium.extent)

The estimated extent of the WRV aquifer system.

```
path <- file.path(dir.in, "extent")
alluvium.extent <- readOGR(path, "alluvium.extent", verbose = FALSE,
                           integer64 = "allow.loss")
alluvium.extent <- spTransform(alluvium.extent, crs)
save(alluvium.extent, file = file.path(dir.out, "alluvium.extent.rda"), compress = "xz")
```

Extent of clay unit (clay.extent)

The estimated extent of the confining hydrogeologic clay unit (aquitard) separating the unconfined aquifer from the underlying confined aquifer in the WRV.

```
path <- file.path(dir.in, "extent")
clay.extent <- readOGR(path, "clay.extent", verbose = FALSE, integer64 = "allow.loss")
clay.extent <- spTransform(clay.extent, crs)
save(clay.extent, file = file.path(dir.out, "clay.extent.rda"), compress = "xz")
```

Extent of basalt (basalt.extent)

The estimated extent of the basalt hydrogeologic unit underlying the alluvial WRV aquifer system.

```
path <- file.path(dir.in, "extent")
basalt.extent <- readOGR(path, "basalt.extent", verbose = FALSE)
basalt.extent <- spTransform(basalt.extent, crs)
save(basalt.extent, file = file.path(dir.out, "basalt.extent.rda"), compress = "xz")
```

Drain boundaries at Stanton Crossing and Silver Creek (drains)

Polygons used to define the location of drain boundary conditions in the model domain. The polygons clip the line segments along the aquifer boundary (see alluvium.extent), and model cells intersecting these clipped-line segments are defined as boundary cells.

```
file <- file.path(dir.in, "drains.kml")
p <- suppressWarnings(readOGR(file, basename(file), verbose = FALSE))
p@data$Description <- NULL
file <- file.path(dir.in, "drains.csv")
d <- read.csv(file, strip.white = TRUE)
p@data <- dplyr::left_join(p@data, d, by = "Name")
p <- spTransform(p, crs)
drains <- p
save(drains, file = file.path(dir.out, "drains.rda"), compress = "xz")
```

Tributary basin underflow (tributaries)

Data that describe the location and average flow conditions for model boundaries in the major tributary canyons of the WRV. Spatial polygons are used to define the boundary locations in the study area.

```
file <- file.path(dir.in, "tributaries.kml")
p <- suppressWarnings(readOGR(file, basename(file), verbose = FALSE))
p <- spTransform(p, crs)
p@data$Description <- NULL
```

For each tributary canyon, the mean rate of tributary basin underflow is first estimated using a Darcian analysis. The lower-half of an ellipse is used to describe the geometry of the saturated cross-sectional area.

```
file <- file.path(dir.in, "tributaries.csv")
d <- read.csv(file, strip.white = TRUE)
names(d) <- sub("_m$", "", names(d))
d$K <- d$K_ft.per.d * ft.to.m
d$K_ft.per.d <- NULL
d$SatArea <- (pi * d$TribWidth * d$BdrkDepth) / 4  # area of lower-half of ellipse
d$DarcyFlow <- d$K * d$SatArea * d$LandGrad
```

A drainage basin area for a tributary canyon is defined as the area of land where groundwater from rain or melting snow converges at the boundary. Smaller basins were identified by plotting basin areas for each tributary canyon on a logarithmic scale (appendix E, fig. E3). A natural break was found at about 26 square-kilometers (10 square-miles) and tributary basins less than this break were designated as “small”, and all others were designated as “big”.

```
d$BasinArea <- d$BasinArea_mi2 * mi2.to.m2
d$BasinArea_mi2 <- NULL
natural.basin.area.break <- 2.59e+7  # 10 square-miles
is.small <- d$BasinArea < natural.basin.area.break
d$BasinAreaType <- as.factor(c("big", "small")[is.small + 1L])
```

Multiplying a basin’s average rate of precipitation by its drainage area gives a secondary estimate of the mean volumetric flow rate at the boundary.

```
d$PrecipRate <- d$PrecipRate_in.per.yr * in.per.y.to.m.per.d
d$PrecipRate_in.per.yr <- NULL
d$PrecipFlow <- d$BasinArea * d$PrecipRate
```

The ratio of Darcy to precipitation volumetric flow rate estimates is then calculated.

```
d$FlowRatio <- d$DarcyFlow / d$PrecipFlow
```

Finally, the mean volumetric flow rate of a small tributary canyon is calculated by multiplying its precipitation flow rate estimate by the average flow rate ratio for the larger tributary canyons.

```
d$Flow <- d$DarcyFlow
d$Flow[is.small] <- d$PrecipFlow[is.small] * mean(d$FlowRatio[!is.small])
```

Save the dataset to disk:

```
p@data <- dplyr::left_join(p@data, d, by = "Name")
tributaries <- p
save(tributaries, file = file.path(dir.out, "tributaries.rda"), compress = "xz")
```

Precipitation zones (precip.zones)

Precipitation zones of the WRV and surrounding areas. Northing coordinate of zone separators is based on the northing midpoint between weather stations.

```
ids <- c("Ketchum", "Hailey", "Picabo")
y <- coordinates(weather.stations)[match(rev(ids), weather.stations@data$id), "y"]
s <- c(mean(y[1:2]), mean(y[2:3]))  # northing zone separators
e <- extend(ext, 5000)  # spatial extent of zones
p1 <- rbind(c(e[1], s[2]), c(e[2], s[2]), c(e[2], e[4]), c(e[1], e[4]))
p2 <- rbind(c(e[1], s[1]), c(e[2], s[1]), c(e[2], s[2]), c(e[1], s[2]))
p3 <- rbind(c(e[1], e[3]), c(e[2], e[3]), c(e[2], s[1]), c(e[1], s[1]))
p1 <- Polygons(list(Polygon(rbind(p1, p1[1, ]))), ID = 1)
p2 <- Polygons(list(Polygon(rbind(p2, p2[1, ]))), ID = 2)
p3 <- Polygons(list(Polygon(rbind(p3, p3[1, ]))), ID = 3)
p <- SpatialPolygons(list(p1, p2, p3), proj4string = crs)
p <- SpatialPolygonsDataFrame(p, data.frame(ID = 1:3, PrecipZone = ids))
precip.zones <- p
save(precip.zones, file = file.path(dir.out, "precip.zones.rda"), compress = "xz")
```

Irrigation entities (irr.entities)

Irrigation entities of the WRV and surrounding areas.

```
path <- file.path(dir.in, "irr")
irr.entities <- readOGR(path, "irr.entities", verbose = FALSE)
irr.entities <- spTransform(irr.entities, crs)
irr.entities <- rgeos::gBuffer(irr.entities, width = 0, byid = TRUE)
d <- irr.entities@data
d$EntitySrce <- as.factor(paste(d$EntityName, d$Source))
d$PrecipZone <- over(rgeos::gCentroid(irr.entities, byid = TRUE), precip.zones)$PrecipZone
irr.entities@data <- d[, c("EntityName", "Source", "EntitySrce", "PrecipZone")]
save(irr.entities, file = file.path(dir.out, "irr.entities.rda"), compress = "xz")
```

Irrigation lands (irr.lands)

Irrigated and semi-irrigated lands of the WRV.

```
path <- file.path(dir.in, "irr")
yr <- c(1996, 2000, 2002, 2006, 2008, 2009, 2010)
files <- paste0("irr.lands.", yr)
irr.lands <- list()
for (i in seq_along(files)) {
  p <- readOGR(path, files[i], verbose = FALSE)
  p <- spTransform(p, crs)
  p@data <- p@data[, paste0("STATUS_", substr(yr[i], 1, 3)), drop = FALSE]
  names(p@data) <- "Status"
  p <- p[p@data[, "Status"] != "non-irrigated", ]
  p <- rgeos::gBuffer(p, width = 0, byid = TRUE)
  p@data <- droplevels(p@data)
  irr.lands[[i]] <- p
}
names(irr.lands) <- as.character(yr)
save(irr.lands, file = file.path(dir.out, "irr.lands.rda"), compress = "xz")
```

Lakes and reservoirs (lakes)

Lakes and reservoirs of the WRV and surrounding areas.

```
path <- file.path(dir.in, "decorative")
lakes <- readOGR(path, "lakes", verbose = FALSE)
lakes <- spTransform(lakes, crs)
save(lakes, file = file.path(dir.out, "lakes.rda"), compress = "xz")
```

Public land parcels (public.parcels)

Non-irrigated public land parcels for areas north of Bellevue.

```
public.parcels <- readOGR(dir.in, "public.parcels", verbose = FALSE)
public.parcels <- spTransform(public.parcels, crs)
public.parcels <- as(public.parcels, "SpatialPolygons")
save(public.parcels, file = file.path(dir.out, "public.parcels.rda"), compress = "xz")
```

Soil Units (soils)

Soil units of the WRV and surrounding areas.

```
soils <- readOGR(dir.in, "soils", verbose = FALSE)
soils <- spTransform(soils, crs)
soils@data <- soils@data[, c("GroupSym", "SoilLayer")]
names(soils@data)[1] <- "GroupSymbol"
file <- file.path(dir.in, "soils.csv")
d <- read.csv(file, strip.white = TRUE)
d$MinRate         <- d$MinRate_ft.per.mo * ft.to.m
d$MaxRate         <- d$MaxRate_ft.per.mo * ft.to.m
d$PercolationRate <- d$Rate_ft.per.mo    * ft.to.m
d <- d[, c("GroupSymbol", "SoilClass", "MinRate", "MaxRate", "PercolationRate")]
d <- suppressWarnings(dplyr::left_join(soils@data, d, by = "GroupSymbol"))
d$GroupSymbol <- as.factor(d$GroupSymbol)
d$PercolationRate[is.na(d$PercolationRate)] <- 0
soils@data <- d
save(soils, file = file.path(dir.out, "soils.rda"), compress = "xz")
```

Wetlands (wetlands)

Wetlands of the WRV and surrounding areas.

```
wetlands <- readOGR(dir.in, "wetlands", verbose = FALSE)
wetlands <- spTransform(wetlands, crs)
wetlands <- as(wetlands, "SpatialPolygons")
save(wetlands, file = file.path(dir.out, "wetlands.rda"), compress = "xz")
```

Bellevue WWTP ponds (bellevue.wwtp.ponds)

Bellevue Waste Water Treatment Plant ponds.

```
bellevue.wwtp.ponds <- readOGR(dir.in, "bellevue.wwtp.ponds", verbose = FALSE)
bellevue.wwtp.ponds <- spTransform(bellevue.wwtp.ponds, crs)
bellevue.wwtp.ponds <- as(bellevue.wwtp.ponds, "SpatialPolygons")
save(bellevue.wwtp.ponds, file = file.path(dir.out, "bellevue.wwtp.ponds.rda"),
     compress = "xz")
```

Kriging zones (kriging.zones)

Kriging zones used in parameter estimation.

```
path <- file.path(dir.in, "opt")
kriging.zones <- readOGR(path, "kriging.zones", verbose = FALSE)
kriging.zones <- spTransform(kriging.zones, crs)
save(kriging.zones, file = file.path(dir.out, "kriging.zones.rda"), compress = "xz")
```

U.S. State of Idaho (idaho)

Boundary of the U.S. state of Idaho.

```
path <- file.path(dir.in, "decorative")
idaho <- readOGR(path, "idaho", verbose = FALSE)
idaho <- as(idaho, "SpatialPolygons")
idaho <- spTransform(idaho, crs)
save(idaho, file = file.path(dir.out, "idaho.rda"), compress = "xz")
```

Grids

Thickness of the quaternary sediment (alluvium.thickness)

The estimated thickness of the Quaternary sediment in the WRV aquifer system.

```
file <- file.path(dir.in, "alluvium.thickness.tif")
alluvium.thickness <- readGDAL(file, band = 1, silent = TRUE)
alluvium.thickness <- projectRaster(raster(alluvium.thickness), high.res.spatial.grid)
alluvium.thickness <- aggregate(alluvium.thickness, fact = 5L, fun = median)
names(alluvium.thickness) <- "alluvium.thickness"
save(alluvium.thickness, file = file.path(dir.out, "alluvium.thickness.rda"),
     compress = "xz")
```

Topography of land surface (land.surface)

The topography of the land surface in the WRV and vicinity.

```
path <- file.path(dir.in, "grdn44w115_13")
r <- raster(readGDAL(path, band = 1, silent = TRUE))
land.surface <- projectRaster(r, high.res.spatial.grid)
land.surface <- aggregate(land.surface, fact = 5L, fun = median)
land.surface[is.na(alluvium.thickness)] <- NA
names(land.surface) <- "land.surface"
save(land.surface, file = file.path(dir.out, "land.surface.rda"), compress = "xz")
```

Hillshading based on the slope and aspect of land-surface elevations.

```
ext <- extent(spatial.grid)
ext <- extent(c(extendrange(c(ext@xmin, ext@xmax), f = 0.05),
                extendrange(c(ext@ymin, ext@ymax), f = 0.05)))
nrows <- (ext@ymax - ext@ymin) / 20
ncols <- (ext@xmax - ext@xmin) / 20
r <- projectRaster(r, raster(ext, nrows = nrows, ncols = ncols, crs = crs))
r[] <- r[] * 2
r <- hillShade(slope = terrain(r, opt = "slope"), aspect = terrain(r, opt = "aspect"))
r.range <- range(r[], na.rm = TRUE)
r[] <- findInterval(r[], seq(r.range[1], r.range[2], length.out = 255)) / 255
r[] <- round(r[], digits = 6)
hill.shading <- r
save(hill.shading, file = file.path(dir.out, "hill.shading.rda"), compress = "xz")
```

Evapotranspiration (et)

Average monthly evapotranspiration in the WRV and surrounding areas.

```
files <- file.path(dir.in, "et", paste0("et.", yr.mo, ".tif"))
FUN <- function(i) {
  r <- readGDAL(files[i], band = 1, silent = TRUE)
  r[[1]] <- r[[1]] * mm.to.m
  return(r)
}
et.raw <- lapply(seq_along(files), FUN)
names(et.raw) <- as.character(yr.mo)
```

Project the raster data into the model grid, and place an upper and lower limit on evapotranspiration that is 3 standard deviations from the mean value.

```
is.missing <- is.na(alluvium.thickness)
FUN <- function(i) {
  r <- aggregate(projectRaster(raster(i), high.res.spatial.grid), fact = 5L)
  r[is.missing] <- NA
  upper.limit <- mean(r[], na.rm = TRUE) + sd(r[], na.rm = TRUE) * 3
  r[r > upper.limit] <- upper.limit
  return(round(r, digits = 6))
}
et <- stack(lapply(et.raw, FUN), quick = TRUE)
names(et) <- as.character(yr.mo)
save(et, file = file.path(dir.out, "et.rda"), compress = "xz")
```

Level 2 Data

Datasets processed at level 2 result from analysis of level 1 data.

Polygons

Monthly irrigation entity components (entity.components)

Irrigation entities and their components in the WRV and surrounding areas. An irrigation entity is defined as an area served by a group of surface-water and/or groundwater diversion(s). Start with the spatial polygon representing the irrigated lands and remove those areas that are designated as wetlands or public parcels. Intersect the resulting polygon with the irrigation entities and calculate the area for each entity.

```
p <- irr.lands
p <- lapply(p, function(i) inlmisc::SetPolygons(i, wetlands, "gDifference", 0.001))
p <- lapply(p, function(i) inlmisc::SetPolygons(i, public.parcels, "gDifference", 0.001))
p <- lapply(p, function(i) inlmisc::SetPolygons(irr.entities, i, "gIntersection", 0.001))
for (i in seq_along(p)) p[[i]]@data$area <- rgeos::gArea(p[[i]], byid = TRUE)
irr.by.entity <- p
```

Irrigation entities are subdivided by water source; that is, “surface-water only”, “groundwater only”, or “mixed source”. Aggregate irrigation entities by water source and calculate their area.

```
FUN <- function(i) {
  d <- aggregate(i@data$area, by = list(i@data$EntitySrce), sum, na.rm = TRUE)
  names(d) <- c("EntitySrce", "area")
  FUN <- function(j) as.character(i@data$PrecipZone[i@data$EntitySrce == j][1])
  d$PrecipZone <- as.factor(vapply(d$EntitySrce, FUN, ""))
  return(d)
}
area.by.entity <- lapply(irr.by.entity, FUN)
```

Calculate the volumetric components of evapotranspiration, precipitation, and crop irrigation requirement for each irrigation entity and water source.

```
FUN <- function(i) {
  yr <- irr.lands.year$IL_Year[irr.lands.year$Year %in% substr(i, 1, 4)]
  p <- irr.by.entity[[yr]]
  unique.sources <- sort(unique(as.character(p@data$EntitySrce)))
  FUN <- function (j) {
    x <- rgeos::gUnaryUnion(p[p@data$EntitySrce == j, ])@polygons[[1]]
    slot(x, "ID") <- j
    return(x)
  }
  sp <- SpatialPolygons(lapply(unique.sources, FUN), proj4string = crs(et.raw[[i]]))
  p <- over(sp, et.raw[[i]], fn = mean, na.rm = TRUE)
  d <- as.data.frame(list(EntitySrce = rownames(p), mean.et = p[, 1]))
  d <- suppressWarnings(dplyr::left_join(d, area.by.entity[[yr]], by = "EntitySrce"))
  d$et.vol <- d$mean.et * d$area
  d$precip.vol <- NA
  for (j in levels(d$PrecipZone)) {
    is.in.zone <- d$PrecipZone == j
    idx <- which(precipitation$YearMonth == i & precipitation$PrecipZone == j)
    d$precip.vol[is.in.zone] <- d$area[is.in.zone] * precipitation$Precip[idx]
  }
  d$cir.vol <- d$et.vol - d$precip.vol
  idxs <- match(d$EntitySrce, irr.entities@data$EntitySrce)
  d[, c("EntityName", "Source")] <- irr.entities@data[idxs, c("EntityName", "Source")]
  rownames(d) <- d$EntitySrce
```

```
  return(SpatialPolygonsDataFrame(sp, d))
}
entity.components <- lapply(yr.mo, FUN)
names(entity.components) <- yr.mo
save(entity.components, file = file.path(dir.out, "entity.components.rda"),
     compress = "xz")
```

Grids

Rasterized canals (r.canals)

Rasterize canals on model grid.

```
r <- rasterize(canals, land.surface, "EntityName", silent = TRUE)
r <- ratify(r, count = TRUE)
d <- levels(r)[[1]]
d$EntityName <- levels(canals@data$EntityName)[d$ID]
levels(r) <- d
r.canals <- r
save(r.canals, file = file.path(dir.out, "r.canals.rda"), compress = "xz")
```

Rasterized monthly irrigation entities (rs.entities)

Rasterize monthly irrigation entities on model grid.

```
FUN <- function(i) {
  r <- rasterize(entity.components[[i]], land.surface, "EntityName", silent = TRUE)
  r <- ratify(r, count = TRUE)
  d <- levels(r)[[1]]
  d$EntityName <- levels(entity.components[[i]]@data$EntityName)[d$ID]
  levels(r) <- d
  return(r)
}
rs.entities <- stack(lapply(yr.mo, FUN), quick = TRUE)
names(rs.entities) <- yr.mo
save(rs.entities, file = file.path(dir.out, "rs.entities.rda"), compress = "xz")
```

Rasterized monthly recharge on non-irrigated lands (rs.rech.non.irr)

Calculate rasterized monthly volumetric flow rates of areal recharge beneath non-irrigated lands.

```
r.zones <- ratify(rasterize(precip.zones, land.surface, "ID", silent = TRUE))
levels(r.zones) <- cbind(levels(r.zones)[[1]], att = precip.zones@data$PrecipZone)
r.soils <- rasterize(soils, land.surface, "PercolationRate", silent = TRUE)
cell.area <- xres(land.surface) * yres(land.surface)
FUN <- function(i) {
  p <- precipitation[precipitation$YearMonth == i, c("PrecipZone", "Precip")]
  p <- p[match(p$PrecipZone, levels(r.zones)[[1]]$att), "Precip"]
  names(p) <- levels(r.zones)[[1]]$att
  r <- r.zones
  levels(r) <- cbind(levels(r)[[1]], Precip = p)
  r <- deratify(r, "Precip") - et[[i]]
  is.pos <- r > r.soils
  r[is.pos] <- r.soils[is.pos]
  if (i %in% yr.mo.irr) r[!is.na(rs.entities[[i]])] <- NA
  r <- r * cell.area
  r[] <- round(r[], digits = 6)
  return(r)
}
rs.rech.non.irr <- stack(lapply(yr.mo, FUN), quick = TRUE)
names(rs.rech.non.irr) <- yr.mo
save(rs.rech.non.irr, file = file.path(dir.out, "rs.rech.non.irr.rda"), compress = "xz")
```