Universitat Oberta de Catalunya

uoc.edu

Visual EccDNA: An interactive Shiny Dashboard for extrachromosomal circular DNA data output visualization and analysis

Ainhoa García Vicente
Master's degree in Bioinformatics and Biostatistics UOC-UB
Area 3 - Cancer and epigenetics

Marcos Araúzo Bravo (Consultant)
Izaskun Mallona González (Professor responsible)

June 2021

FICHA DEL TRABAJO FINAL

Título del trabajo:	*Visual EccDNA: An interactive Shiny Dashboard for extrachromosomal circular DNA data output visualization and analysis*
Nombre del autor:	*Ainhoa García Vicente*
Nombre del consultor/a:	*Marcos Araúzo Bravo*
Nombre del PRA:	*Izaskun Mallona González*
Fecha de entrega:	*06/2021*
Titulación:	*Máster universitario en Bioinformática y bioestadística UOC-UB*
Área del Trabajo Final:	*TFM-Bioinformática y Bioestadística Area 3*
Idioma del trabajo:	*Inglés*
Número de créditos:	*15 ECTS*
Palabras clave	*eccDNA* *Oncogenes* *Data visualization*

Resumen del Trabajo:

Es sabido desde hace tiempo que algunas células cancerosas presentan pequeñas partículas de material genético fuera de los cromosomas, lo que se conoce como DNA circular extracromosómico (eccDNA). Durante estos últimos años y gracias a la combinación de métodos de purificación y amplificación del DNA, y con el uso de métodos computacionales para su posterior análisis, se ha logrado caracterizar estos círculos, descubriendo muchas veces genes implicados en el crecimiento tumoral y en la resistencia a medicamentos. Cada vez queda más patente la relación entre la presencia de eccDNA y los cánceres con peor pronóstico.

En este ámbito, el doctor Marcos Arauzo y su equipo de biología computacional en el Instituto de Investigación Médica Biodonostia, se encuentran en un programa de colaboración europeo intentando mejorar la detección temprana de eccDNA en pacientes oncológicos, caracterizar los oncogenes componentes de estos círculos así como cuantificar el grado de

sobreexpresión de los mismos. Se trata de un proyecto multidisciplinar, por lo que el motivo de este trabajo final de máster surge de la necesidad de interpretación de los resultados obtenidos por los distintos grupos trabajando en este proyecto.

En este trabajo se detalla el desarrollo de Visual EccDNA, una aplicación basada en R y la librería Shiny, que partiendo de los archivos obtenidos tras el mapeado de secuencias de eccDNA, permite visualizar y navegar por los resultados una manera dinámica e interactiva.

Abstract:

It is known from decades ago that some cancerous cells show little spots of genomic material outside of their chromosomes, referred to as extrachromosomal circular DNA (eccDNA). During the last years and thanks to the combination of wet lab techniques such as purification and amplification of DNA circles, and the development of methods for further computational analysis, it has been possible to identify the genomic composition of these circles, finding out genes involved in tumor growth and drug resistance. There is currently an increasing concern about the connection between eccDNA and cancers with worst prognostics.

Dr Marcos Araúzo and their team of computational biology in the Biodonostia Health Institute Research are involved in an European collaboration project working on early detection of eccDNA in oncologic patients, improving the identification of the oncogenes contained in the circles and also quantifying the increased expression. It consists of a multidisciplinary group of researchers, so the motivation of my master's final project comes from the necessity of easy understanding and interpretation of data output.

In this work it is detailed the development of Visual EccDNA, a dashboard for data visualization based on R language and its package Shiny, that uses mapped data from eccDNA sequences and allows the user to explore and navigate through the results in a dynamic and interactive way, which will help researchers of different areas to better understand the biological information in the circles.

Index

List of figures

List of tables

1. Summary

It is known from decades ago that some cancerous cells show little spots of genomic material outside of their chromosomes, referred to as extrachromosomal circular DNA (eccDNA). During the last years and thanks to the combination of wet lab techniques such as purification and amplification of DNA circles, and the development of methods for further computational analysis, it has been possible to identify the genomic composition of these circles, finding out genes involved in tumor growth and drug resistance. There is currently a increasing concern about the connection between eccDNA and cancers with worst prognostics.[1]

In this work it is detailed the development of Visual EccDNA, a dashboard for data visualization based on R language and its package Shiny, that uses mapped data from eccDNA sequences and allows the user to explore and navigate throw the results in a dynamic and interactive way, which will help researchers of different areas to better understand the biological information in the circles.

2. Introduction

2.1 - Context and justification of the project

In recent years, recent technological advances have enriched our knowledge of eccDNA biology.as the advent of gene sequencing made possible the detection and characterization of the genes composing eccDNA in tumors.

Dr Marcos Araúzo and their team of computational biology in the Biodonostia Health Institute Research [2] are involved in an European collaboration project working on early detection of eccDNA in oncologic patients, improving the identification of the oncogenes contained in the circles and also quantifying the increased expression.

They are trying different computational pipelines to improve the process of mapping sequences of eccDNA. The output of their pipelines leads to BED files, which is the main data I will be working with.

The motivation of my master's final project comes from the necessity of easy understanding and interpretation of data to different teams working in this European project. For that aim navigation and visualization tools are required to identify the genes and possible pathways enriched in the circles. Visual eccDNA will help data interpretation to different research groups.

2.2 - Project's objective

The main objective of this master's final project is to develop a Shiny app for eccDNA data visualization.

- ☐ Objective 1: eccDNA visualization

  Design and develop a suitable way to show genomic information from mapped eccDNA.

  - Process the output files from mapping.
  - Overall data output visualization.
  - Circular plot for each sample showing genomic information.

- ☐ Objective 2 - R/Shiny app

  Develop Shiny app to explore the information in a dynamic and interactive way.

  - Make it possible to upload external output BED files.
  - Design a suitable layout.
  - Easy to understand results and navigate through them (click-in plot)

2.3 - Approach and methods

This project is a custom way of visualizing the particular genomic data contained in eccDNA. The chosen programming language to develop this project is R, because it has the core packages needed to achieve a great result:

- ➔ Bioconductor: provides tools for the analysis and comprehension of high-throughput genomic data.
- ➔ Ggplot2/Plotly: Plotting packages for showing data in a comprehensive and interactive way.
- ➔ Shiny/Shiny Dashboard: facilitates the development of an interactive web app.

Other programming grammar taken into consideration for this purpose are HTML with the help of CSS. This involved creating a SQL server to host and process the genomic information. With the time available for the project, taking into consideration my programming skills within different languages, it was decided that R is the best choice.

The project development has been hosted on GitHub (https://github.com/agarcia18/eccDNA) for version control and to be accessible for everyone when it is finished.

2.4 - Working plan: Milestones and risks analysis

Milestones

Following the deadlines given by the master's teaching plan and the objectives in the project, I designed a working schedule. If there is any delay with the achievement of these milestones, the rest of the activities in the project will be delayed. The most important and priority milestone in the planning is to have a beta app working by the submission of PEC2. This means the code in R processing the information in the files and the Shiny app must work together. Afterwards, better graphic designs and different information layers will be made and added into the app.

When the working on the project began, more time was needed in the process of making the layout and overall display of the information, as well as the correct treatment of the files and page interactivity.

This delayed the design of the circle graph and the genomic information, but it was necessary. Shiny dashboard code works differently when you just show the output of the data hosted in your computer or when it is uploaded from third parties.

Due to my inexperience in this field, this was planned for the end of the project. Luckily, taking this into consideration at the beginning, resulted in a more robust app, in contrast to having to adapt the code for the uploaded data by the end of the project.

PEC 1 – Working plan design (01/03/21 – 16/03/21)

- Choose methods
- Task division and workflow design
- Get in touch with the biological content of the data

PEC 2 – Work development I (17/03/21 – 19/04/21)

31/03/21 Milestone 1: Data process

- Process BED files
- Overall data analysis and quality control
- Overall data table and visualization

19/04/21 Milestone 3: Shiny App (basic)

- Define basic user interface
  - Make possible to upload external files
- Define basic server function
- Integration of the R code in Shiny App (*)
- Check packages dependencies and create the final R package (*)

(*) These two tasks will be carried out during the whole process of the project to make sure everything is working correctly.

PEC 3 – Work development I I (20/04/21 – 17/05/21)

26/04/21 Milestone 4: Finish all data visualization

- Finish circle graph
- Incorporate all relevant genomic information

17/05/21 Milestone 5: Expand Shiny App

- Navigate through different results pages
- Interactive plots

17/05/21 Milestone 6: Upload Shiny App

- Deploy and launch

PEC4 – Milestone 7: Finish writing report (18/05/21 – 08/06/21)

PEC5 – Milestone 8: Defense (June 2021)

13/06/21 Make presentation video and slides

This working schedule was designed in the following Gannt's diagram:

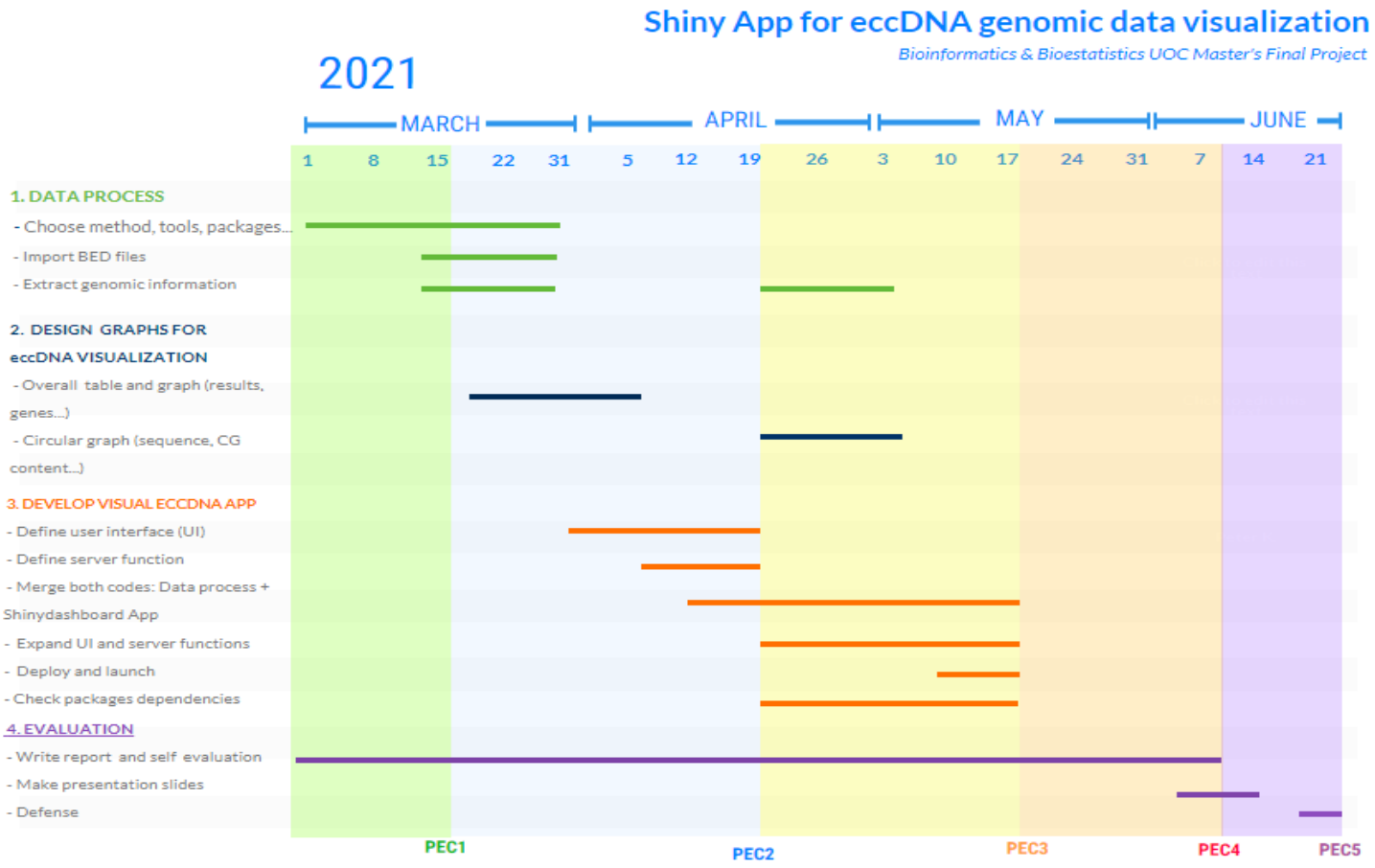


Fig1. Gannt's diagram with working plan for Visual eccDNA app development

Risks analysis

Every project is influenced by the management of time. Being serious about the working schedule planned for this report, following it and making sure to complete all the milestones in time is the best way to minimize the project failure.

Another main risk of this project is the particular and unique genomic data I will be working with. The genomic information contained in eccDNA is full of discordant and split reads so getting whole gene sequences in the circles is very complex. To improve the obtention of biological relevant results, I tried to incorporate some extra data obtained from public datasets, but this does not show the reality of the genomic information in eccDNA.

Another disadvantage of the project is that the European collaboration in IIS Biodonostia has just started, so in the time I will be working with them, the access to the real data from cancer patients will not be possible, but I hope this app will help in further work in the future.

2.5 - Brief summary of contributions and obtained products

- Work plan (**PEC1**)
- Products: (**PEC2** and **PEC3**)
  - Visual eccDNA - genomic data visualization Shiny App
  - GitHub repository with project's code
- Written report: Methods, development, results and biological discuss (**PEC4**)
- Final presentation and defense (**PEC5**)

2.6 - Brief description of the contents in this report

Extrachromosomal circular DNA

- What is eccDNA? (genomic characteristics)
- Role in cancer
- Bioinformatics for eccDNA characterization (sequencing and mapping)

Visual eccDNA app development

- Methods: source data, design and development
- Results: performance of the app
- Discussion: biological explanation of the results
- Conclusions: thoughts about the project development

Glossary of terms, references and annexes

3. Extrachromosomal circular DNA (eccDNA)

3.1 - What is eccDNA?

In 1965, researchers found little pieces of DNA floating around chromosomes in cancer cells [3]. They gave them the name of double minutes, double because the dots were found in pairs, and minute because they were minuscule. Today, scientists think that they were likely just DNA circles caught in the act of replicating. Southern blots determined that eccDNA molecules were homologous to genomic DNA. The majority of eccDNA were < 500 base pairs and were renamed poly-disperse circular DNA (spcDNA). [4]

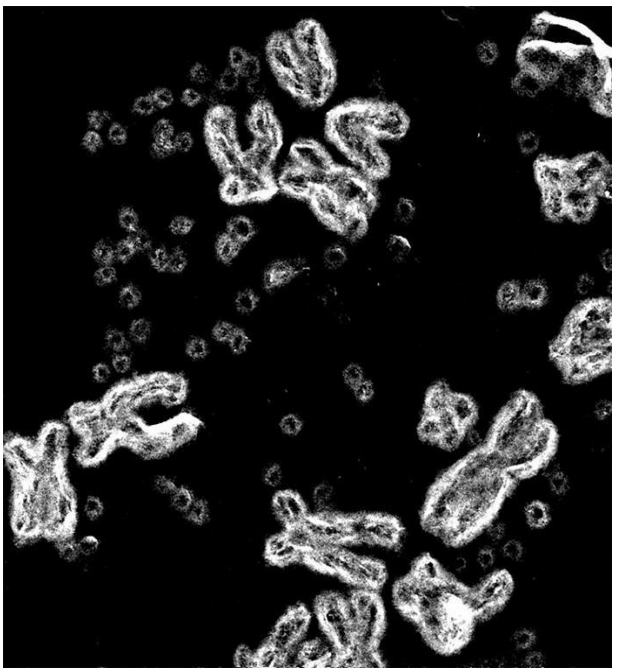


Fig 2. Circles of DNA around chromosomes in samples of colorectal cancer cells in SEM (P.Mischel et.al (c&en) [5]

At that time the field suffered from a technological bottleneck and there were not reliable methods to properly characterize eccDNA. The development of Whole Genome Sequencing (WGS) has allowed researchers to understand the origins, the structural composition and the biological importance of eccDNA. This includes not only to isolate, amplify and sequence these circles, but also the development of bioinformatics tools to map eccDNA.

Early discoveries have related the presence of eccDNA with genomic instability, mismatch repair pathways and transcriptional activity [6]. In contrast to bacterial plasmids or mitochondrial DNA, eccDNA are chromatinized, containing high levels of active histone marks, but a paucity of repressive histone marks. The eccDNA chromatin architecture lacks the higher-order compaction that is present on chromosomal DNA and is among the most accessible DNA in the entire genome.

3.2 - Role in cancer

It seems that different types of eccDNA possess distinct functions. This depends priorly how the circle is formed. Currently the formation of eccDNA is being studied.

There are 4 fundamental hypothesis:

- The translocation-deletion-amplification model [7]
- Chromothripsis [8]
- The breakage-fusion-bridge (BFB) model [9]
- The episome model [10]

Understanding the biological function of eccDNAs will lead to elucidating epigenetic mechanisms behind gene regulation under normal and pathological circumstances. Depending on which genes are amplified can grant cells with advantages or disadvantages.

However, in cancer cells, the role of eccDNA is related to bad pronostics. Because short eccDNAs are poorly chromatinized, is an important mechanism responsible for the amplification of oncogenes. [11]

Aberrant enhancer activity is a key driver of gene expression programs that contribute to tumor formation, maintenance, and progression. Patients whose tumors contained ecDNA amplification had significantly worse five year survival outcomes compared to patients whose tumors harbored either non-circular or no amplifications, demonstrating that the presence of ecDNA is associated with tumor aggressiveness. Circular amplicons are much more prevalent in aggressive cancers such as glioblastoma.[1]

Due to the loose chromatin characteristics of eccDNA, ATAC-seq (assay for transposase-accessible chromatin using sequencing) has been a good approach identifying eccDNA in cancer and cell lines without any enrichment of circular DNA [12].

Cancer cells harboring increased oncogene expression and copy number acquire a competitive advantage and possess the ability to rapidly fit changing environments. This leads to tumor heterogeneity and drug resistance [13] [14].

Extracellular free eccDNAs can also be detected in cancer tissues and blood samples of cancer patients [15]. Therefore, it has been postulated that eccDNAs, especially microDNAs, could be used as a novel type of biomarkers for cancer detection, treatment assessment and prognostic surveillance.

However, eccDNAs have also been detected in healthy tissue [16], so it is more important to analyze the genomic content of the circles to investigate their biological consequences.

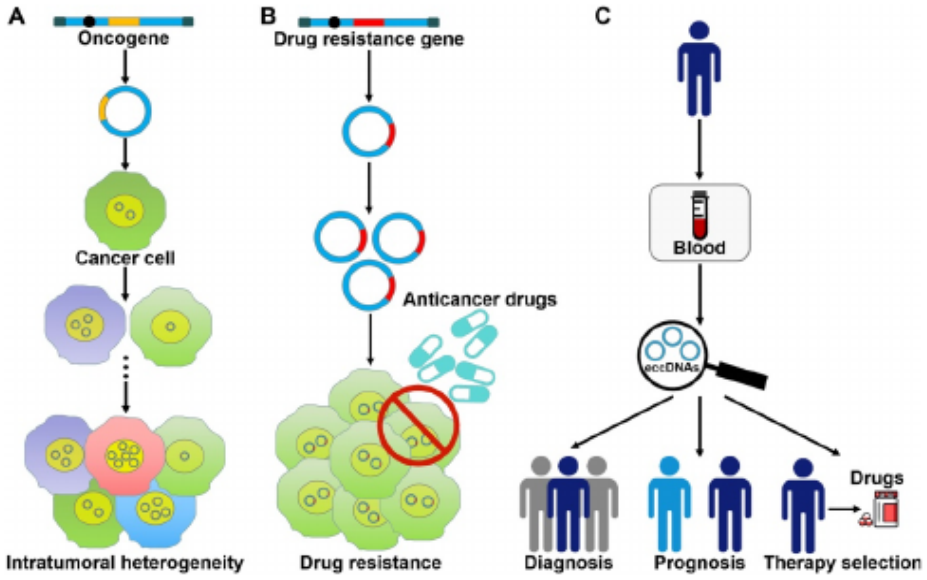


Fig 3. The roles of eccDNAs in cancer pathogenesis. (Man Wang et. al [13])

3.3 - Bioinformatics for eccDNA characterization (Circle-Seq)

In the recent years the development of Whole Genome Sequencing (WGS) methods, in combination with techniques of isolation and amplification of the circles, the preparation of a suitable library and with the use of bioinformatics tools for the analysis of high-throughput genomic information contained in the circles, has made possible the in-depth study of eccDNA leading to the conclusions seen in the previous paragraph. This process is known as Circle-seq.

The experiment must begin with the separation of chromosomal DNA and eccDNA and amplifying the circles for better detection.

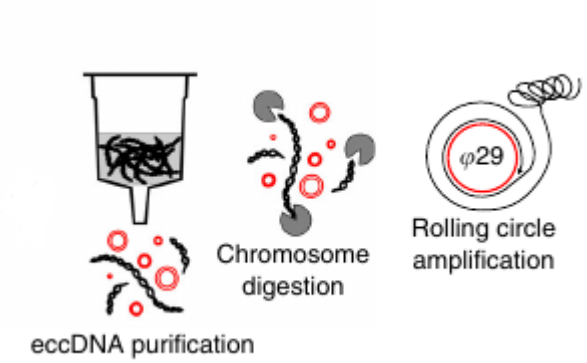


Fig 4. Process of eccDNA purification and amplification (Henrik Devitt Møller et. al [16])

The best approach for eccDNA sequencing, is the use of paired-end sequencing which facilitates the detection of genomic rearrangements and repetitive sequence elements, as well as gene fusions and novel transcripts. Generated from the genome the circle would possess a start and an end position, which were ligated to form a junctional site.

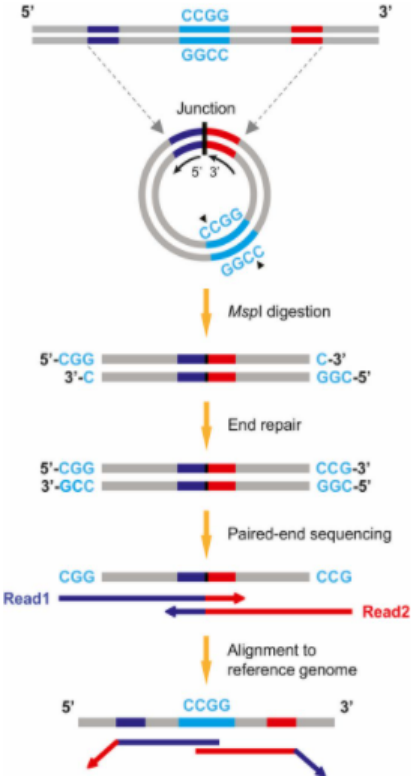


Fig 5. Workflow of eccDNA identification (Sarah T. K. Sin et. al [17])

For the step of mapping, usual tools are used, such as Bowtie2, Hisat2, Bwa-Aln or bwa-mem. Sometimes to improve the results, mapping tools can be adapted to process raw data [16].

The key importance for eccDNA detection in mapped short read data is:

- discordantly mapped paired-end reads
- split-reads crossing the circle breakpoint
- increased mapping coverage over the chromosomal reference genome region of excision

For this purpose, some tools have been developed specifically for circles detection:

- ➔ Circle_finder (https://github.com/pk7zuva/Circle_finder) which was developed for circle detection in combination with ATAC-seq [12].
- ➔ CIRCexplorer2 [18] which has been ranked as one of the best choices for circular RNA detection.
- ➔ Circle-map (https://github.com/iprada/Circle-Map) to accurately identify circular DNA breakpoints. [19]

This last tool was used to obtain the baseline data for Visual eccDNA app development. The following lines include a brief outline of the alignment strategy of Circle-map for eccDNA detection.

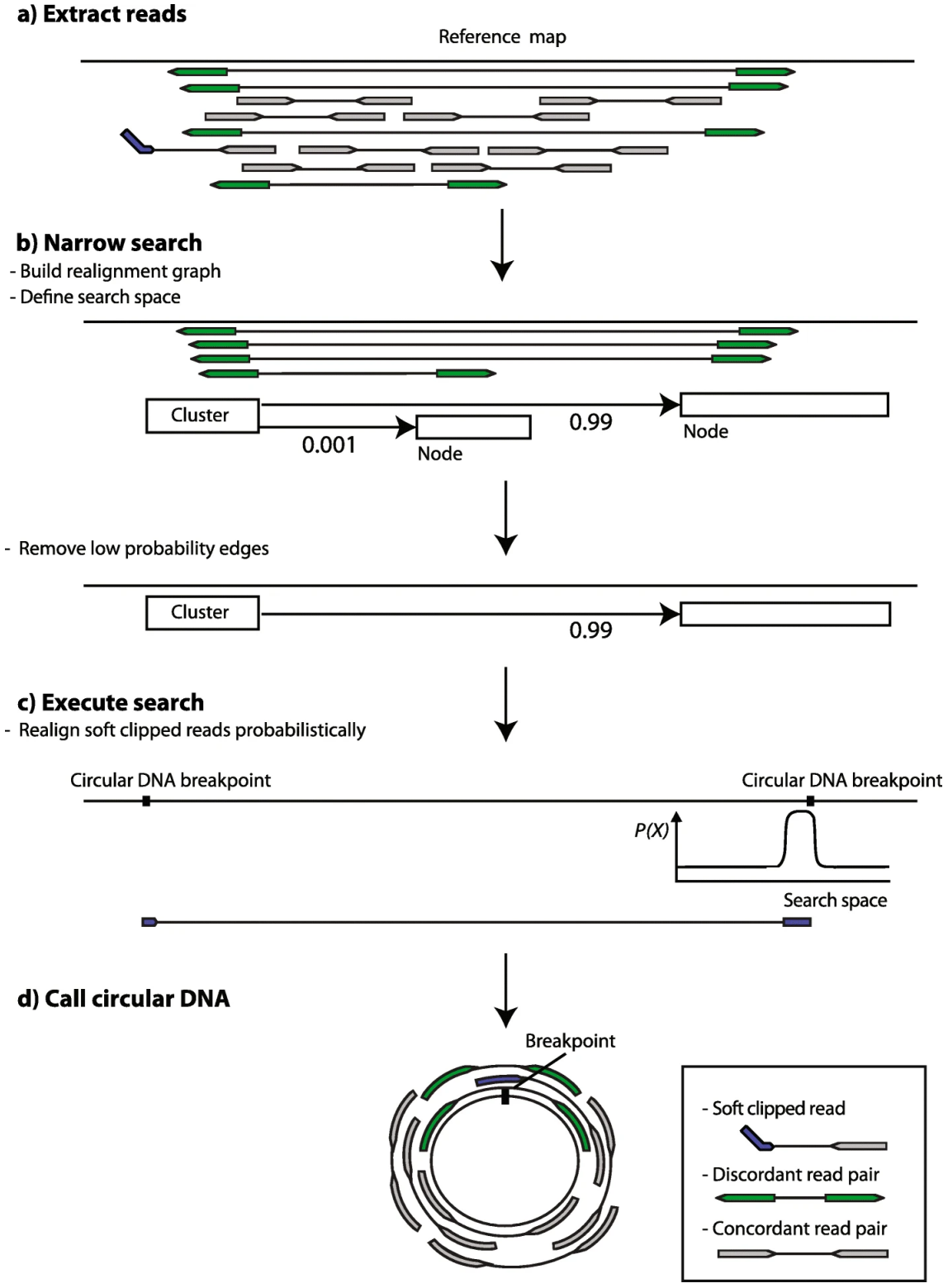


Fig 6. Circle-Map read realignment strategy (Iñigo Prada-Luengo et. al [19])

a. Reads are mapped to the reference genome and discordantly aligned reads (green) and alignments containing soft clips (blue) are extracted; concordantly aligned reads (grey) are ignored.

b. Using the extracted reads, a graph of putative breakpoint connections between genomic regions is constructed and used as a prior to narrow down the genomic search space for realigning soft clipped reads.

c. Non-aligned parts of the soft clipped reads are realigned probabilistically using the breakpoint graph as a guide.

d. Evidence from split-reads and discordant reads are combined to create the final circle calls together with information about concordant, split-read and discordant read coverage for each circle.

Visual eccDNA

Shiny Dashboard for eccDNA data output visualization and analysis

4. Methods

4.1 - Version control (Git & GitHub)

Prior to starting with the development of the project, I installed a version control system (VCS) which allows to record changes to a file or set of files over time so that you can recall specific versions later.

Git is the most popular VCS. It is free and easy to use. When it is installed on your local system, it gives you a self-contained record of your ongoing programming versions. GitHub is designed as a Git repository hosting service. Through GitHub, you can share your code with others, giving them the power to make revisions or edits on your various Git branches. This makes it possible for entire teams to coordinate together on single projects in real-time. [20]

In this project I was working on my own, but it was helpful for keeping track of the changes and for sharing the progress with the advisors. The GitHub repository created to host Visual eccDNA app development online was https://github.com/agarcia18/eccDNA.

After Git is installed, the GitHub account is set and connected with Git, and the repository is created in GitHub, the first step is to clone the repository into the computer to start version control in a folder (`git clone "github repository URL"`). Then, using the same commands you can manage the changes made in your computer and send them into the GitHub repository:

- `git add`: saves changes in files for commit.
- `git commit`: once the files are added, the changes made will be saved as a version of the repository, and are now ready to go up on GitHub. A comment must always be added.
- `git push`: pushes the changes into GitHub.The local copy and the online copy of the files will be the same.

It is important to get into the habit of using `git pull` always before start editing and before push, specially when working with other people. This command makes sure the files in the computer are completely up to date with the latest version of the online version taking care of the changes that could have been made.

4.2 Data

By the time I was involved in this project with Araúzo's team, their collaboration with the European team had just started, so the access to real data from oncologic patients was not possible. The sequence data used for the development of Visual eccDNA is part of an experiment of Circle-seq for detection of eccDNA in skeletal muscle biopsies and blood leukocytes from two groups of healthy men [16]. The circles were previously isolated and amplified. It is deposited in the Sequence Read Archive (SRA) (accession: **SRX3415277**).

Circle-Map[19] tool was used on this data to detect eccDNA. The output consists of a tab-separated BED file containing 11 fields with all the detected circular DNA, containing mapping and information about the sequencing coverage. BED (Browser Extensible Data) format is a text file used to store genomic regions as coordinates and associated annotations separated by tabs.

COLUMN	FIELD
1	Chromosome
2	Start coordinate
3	End coordinate
4	Discordant reads
5	Split reads
6	Circle score
7	Mean coverage
8	Standard deviation
9	Coverage increase in the start coordinate
10	Coverage increase in the end coordinate
11	Coverage continuity

Table 1. Output of Circle-map BED files [23]

Unfortunately, due to the particular characteristics of the genomic data in the circles, added to my inexperience in processing NGS data and the time given for the development of the master's final project, it was not possible for me to get involved more in the processing of the raw data. I started the project with preprocessed data by Circle-map, but Visual eccDNA can be improved in the future to help share results of different Circle-seq experiments and also compare the output of different mapping and circle detecting bioinformatics tools.

4.3 Overall design and development

Visual eccDNA is developed using R language, with **Shiny** open source package, which allows the creation of web applications. [21]

Shiny allows to build apps with interactivity (elements that react to user actions, f.ex: zoom, click-in plots) and reactivity (data shown is updated from the server without refreshing the site, f.ex: filtering dataset).

Shiny apps are contained in a single script called `app.R`, that has three components:

- User interface object (UI): controls the layout and appearance of the app
- Server function: contains the instructions that your computer needs to build your app
- A call to the shinyApp function: creates Shiny app objects from an explicit UI/server pair.

Shiny Dashboard is an extension of Shiny to make dashboards [22], a graphical interface for users to quickly visualize important metrics. It provides a professional looking product without needing to know HTML or CSS.

As this app is thought to be shared between different labs, the external upload of Circle-map BED files needed to be the first step. It begins empty, but with the use of `uiOutput` in the UI and `renderUI` in the server, is being built step by step.

The app is divided in the following tabs:

- ☐ Get started - Upload BED files and get an overall results of the file
- ☐ Small/Big circles - Shows circles by size and chromosome, genes and pathways
- ☐ Circle info - Base content and genes in each circle

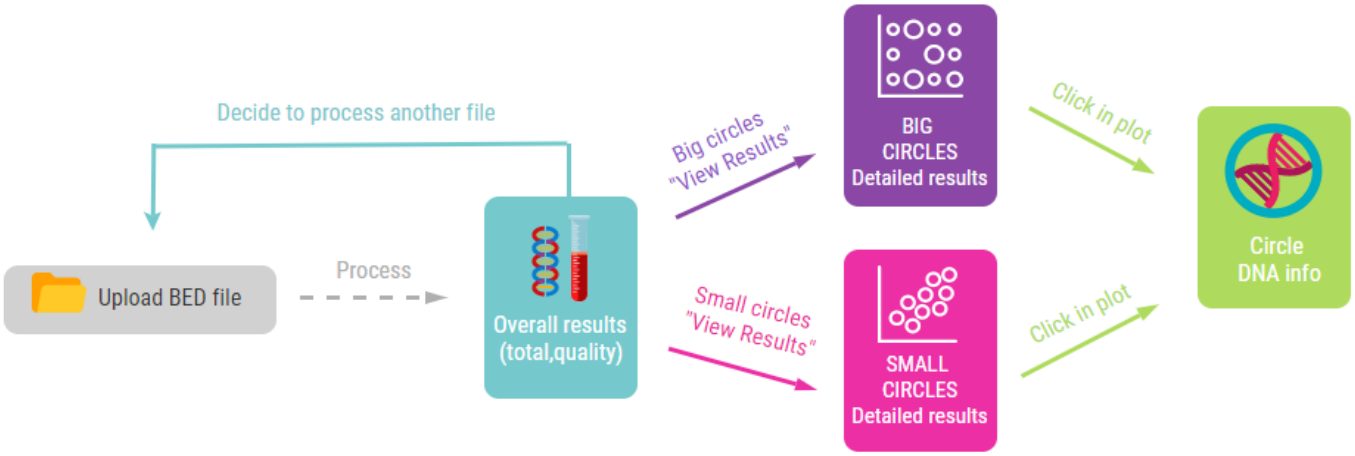


Fig 7. Visual eccDNA app structure design

4.3.1.“Get Started” tab

This is the only part of the app that shows when it is loaded for the first time. It contains:

- Banner and logo
- Brief explanation of how the app works
- File input to load Circle-map output BED files for process.

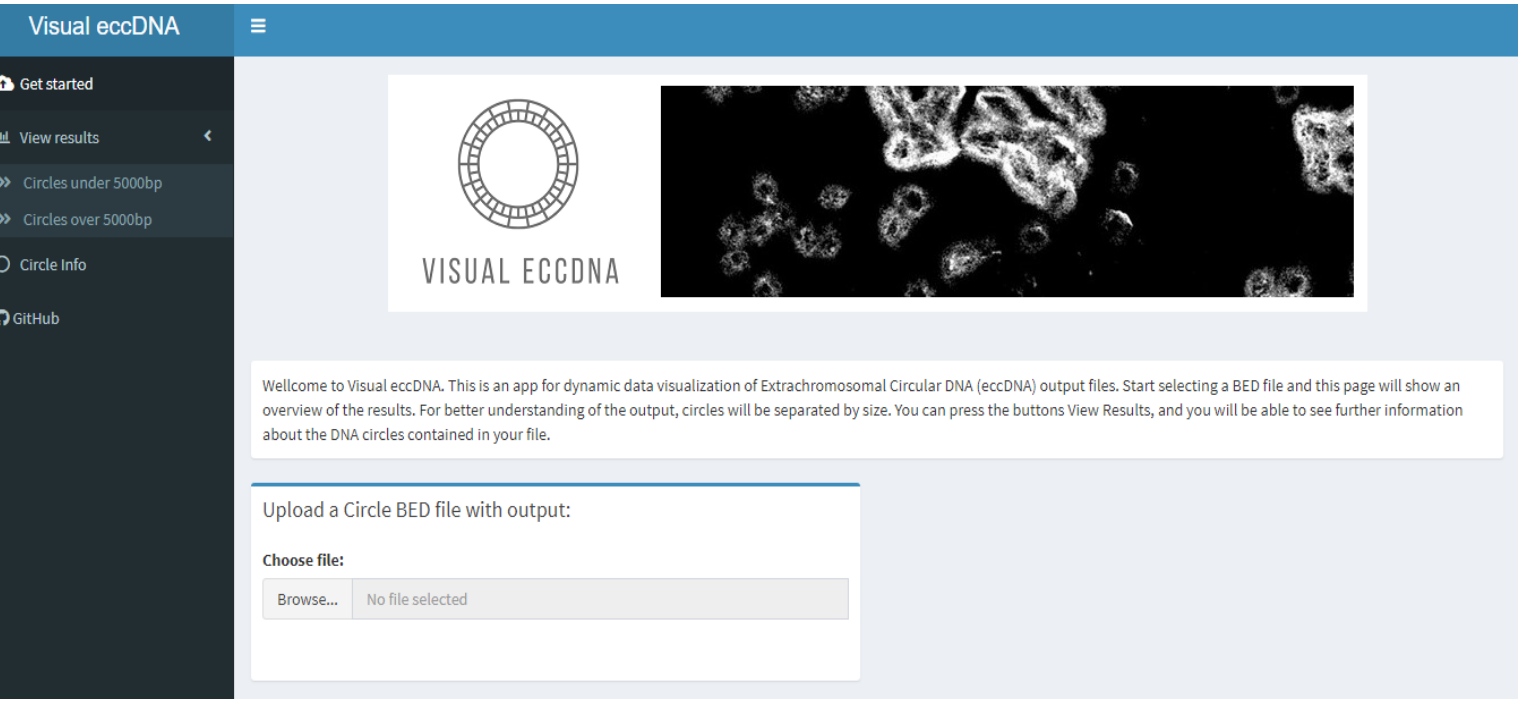


Fig 8. “Get started” tab when the app is just loaded

After the file is uploaded, the server processes it, setting the data as a data frame, adding the names of the columns for each field (as shown in Table 1) and a circle ID number. Based on Circle-map wiki [23] for the circle score field, a quality tag was also included: Bad (< 10), Low (10-50), Medium (50-200), Good (> 200). Finally, also the size in base pairs (bp) was calculated with the difference between the coordinates.

Having the size of the circles, the percentage of small and big ones within the file is calculated (under and over 5000bp). This number was decided after checking between the different files and observing that most of the eccDNA (aprox 85%) lay below this checkpoint. If needed, this number can be easily changed in the beginning of the code of the app. This split of the results in that manner helps to visualize the data in a better way.

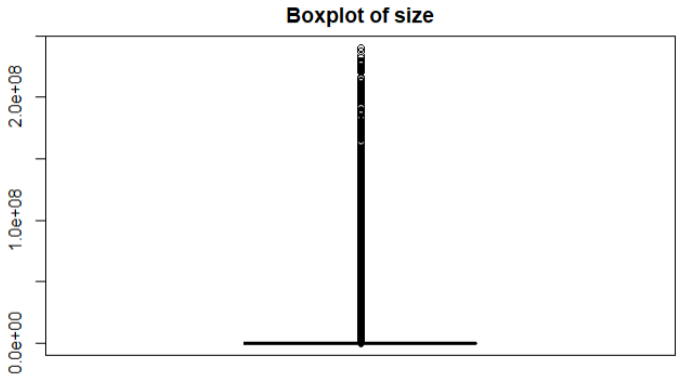
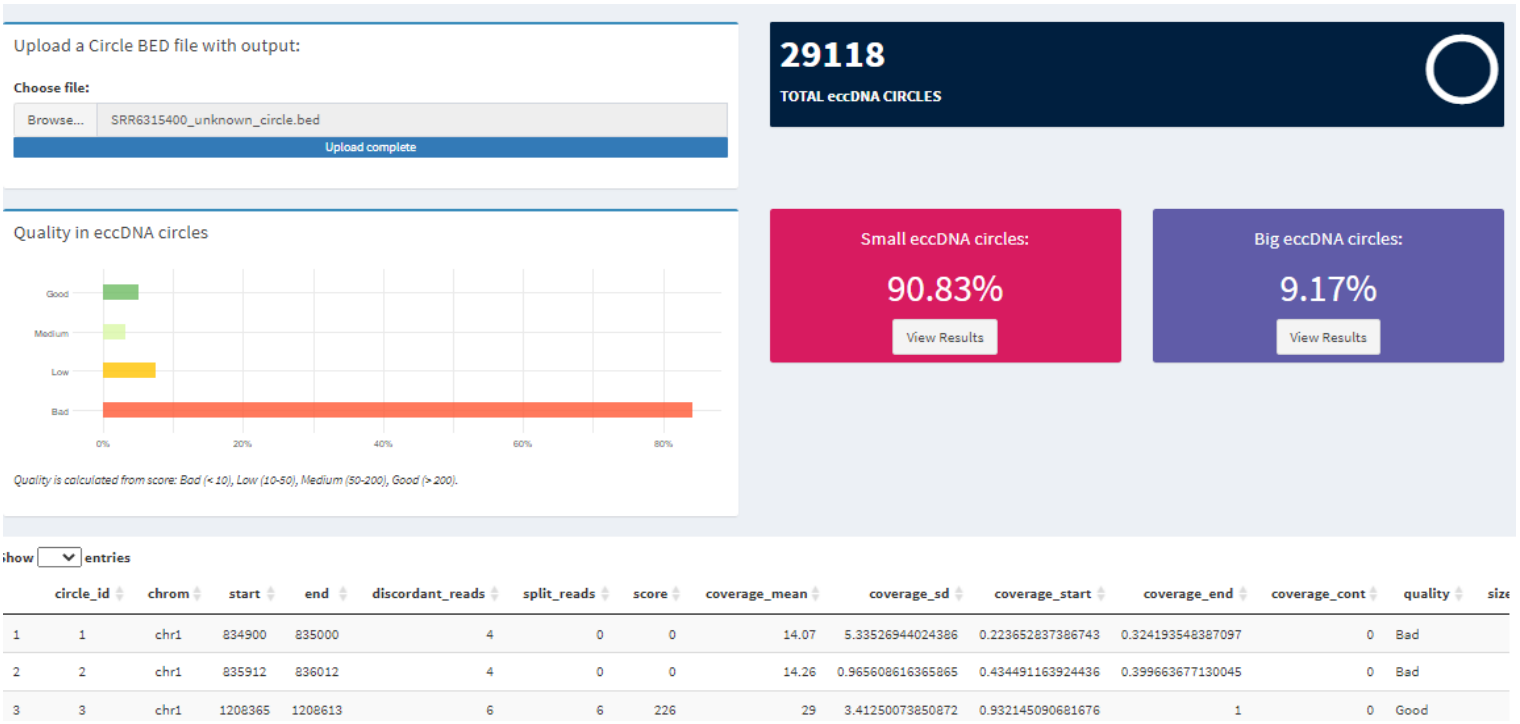


Fig 9. The problem of size distribution

So once the desired BED file is uploaded correctly the pages reacts and also shows:

- Number of total eccDNA circles
- Quality plot
- Percentage of small and big eccDNA circles
- Table output with the uploaded file (with circle ID, size and quality added)



	circle_id	chrom	start	end	discordant_reads	split_reads	score	coverage_mean	coverage_sd	coverage_start	coverage_end	coverage_cont	quality	size
1	1	chr1	834900	835000	4	0	0	14.07	5.33526944024386	0.223652837386743	0.324193548387097	0	Bad	
2	2	chr1	835912	836012	4	0	0	14.26	0.965608616365865	0.434491163924436	0.399663677130045	0	Bad	
3	3	chr1	1208365	1208613	6	6	226	29	3.41250073850872	0.932145090681676	1	0	Good	

Fig 10. "Get Started" tab after uploading a file

For data manipulation, **dplyr** package was used. It provides simple "verbs", functions that correspond to the most common data manipulation tasks, to help translate thoughts into code. By constraining the options, it helps you think about your data manipulation challenges and it also uses efficient backends, to process the data rapidly. [24]

ggplot2 is also a package used for the development of many of the plots shown in the app. The layered grammar of graphics approach is implemented. All graphics in this library are built using a layered approach, building layers up to create the final graphic.[25] Sometimes for character manipulation of the labels in the plot, **stringr** [26] package was also used.

The tables in the app were built with the usage of **DT** [27]**,** which provides an R interface to the JavaScript library DataTables. R data objects (matrices or data frames) can be displayed as tables on HTML pages, and DataTables(DT) provides filtering, pagination, sorting, and many other features in the tables.

To advance to the next part of the app, “View Results”, `actionButton` reactivity was implemented. It is also possible to click in the lateral bar.

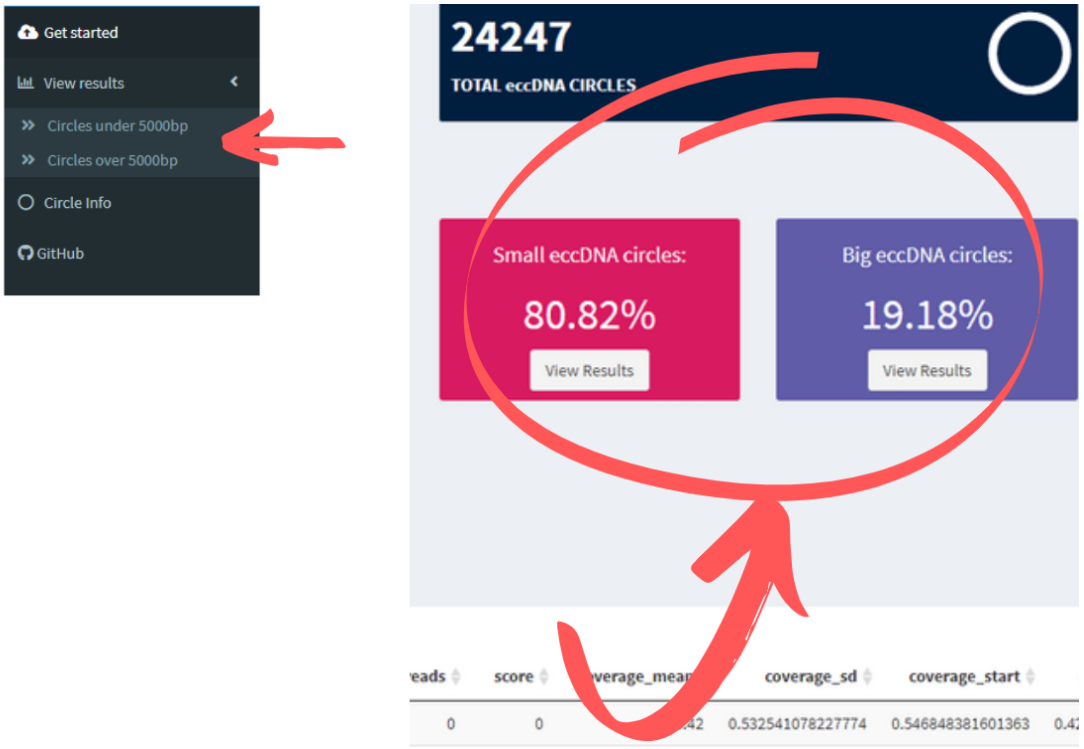


Fig 11. Reactivity to change within tabs

4.3.2. “View results” tab – Circles under/over 5000bp

Both of this tabs show the same information contained in 4 boxes:

- Dynamic plot – Clickable and with hover effects
- Filtering widgets for the plot – Size and quality of the circles
- Genes table
- Enriched pathways of the genes

To facilitate the comprehension of the output in a glance, the styling of each of the outputs is distinguished by size and color (pink for smaller circles and purple for bigger ones).

There is also a “Back to Get Started” `actionButton` to go back to the first output page (it is also possible to click in the lateral panel).

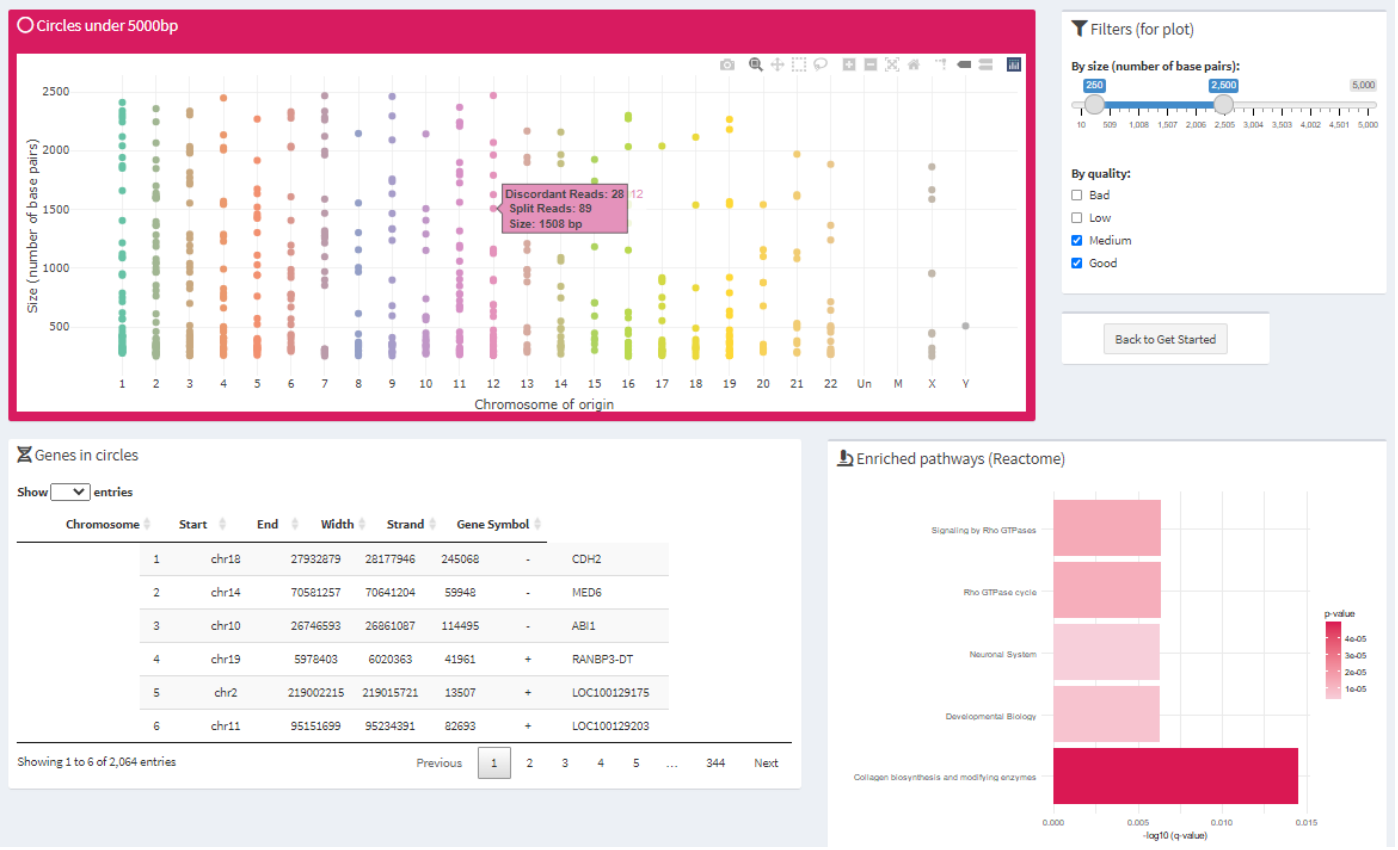


Fig 12. “View results” tab - Circles under 5000bp

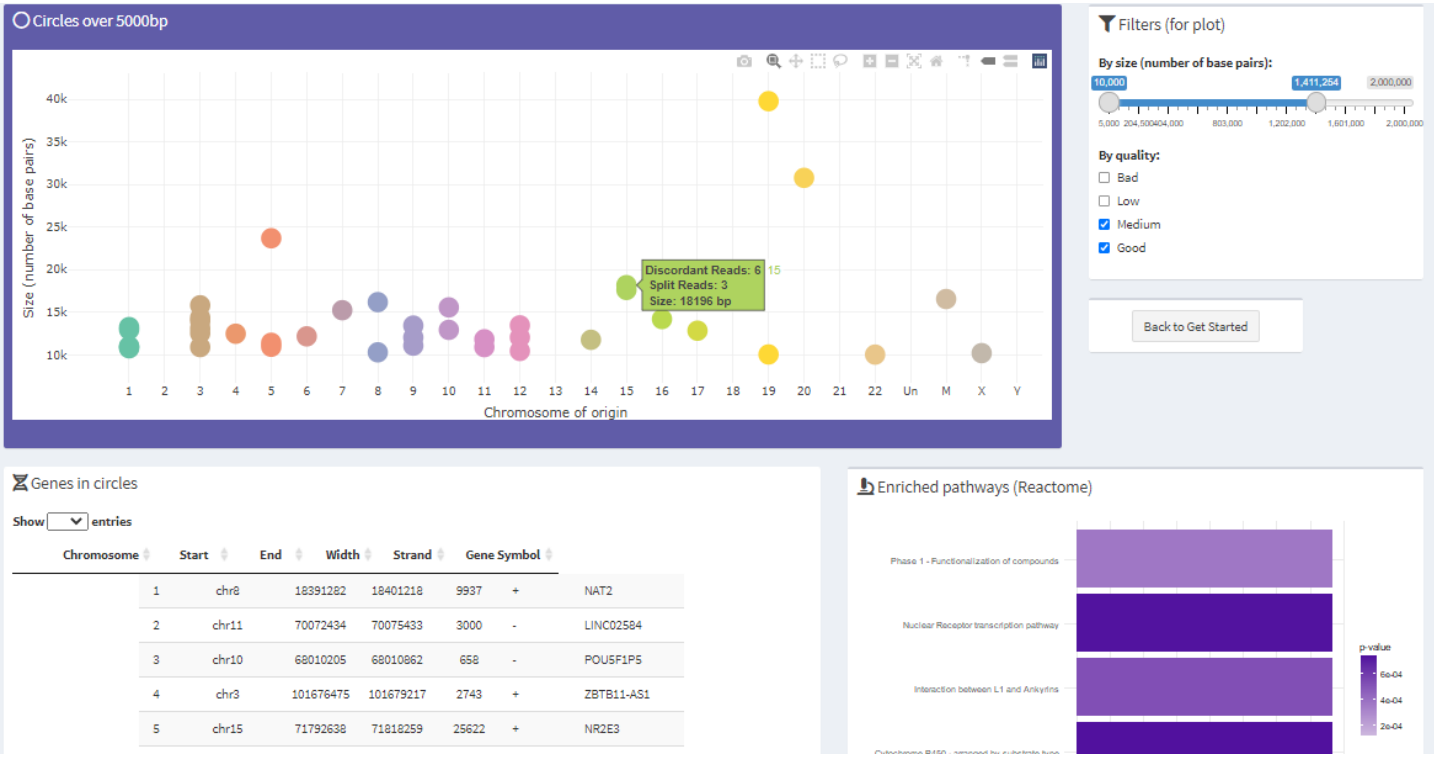


Fig 13. "View results" tab - Circles over 5000bp

The main plot shows the circles in the file, by chromosome and size. It was designed using **plotly** [28] because this package allows to implement reactivity in the plot. It is possible to select, zoom in or out, and the hovering effect shows more information about a circle, such as discordant reads and split reads. It also has a click on reactive effect that gives the user access to the last part of the app, the individual eccDNA genomic information.

Filtering widgets `sliderInput` for size and `checkboxgroupInput` for quality, fixes the problem of disparity in circles, making it possible to see the results for each category properly. The limit of size visualization is 2000000bp. This limit was decided after checking that in most of the files the biggest circles outliers have bad quality. This value can also be changed at the beginning of the code.

It is possible to use **Bioconductor** project packages within Shiny for the genomic interpretation of the data. Bioconductor is a free, open source and open development software project for the analysis and comprehension of genomic data software repository that hosts a wide range of statistical tools developed in the R programming environment. Utilizing a rich array of statistical and graphical features in R, many Bioconductor packages have been developed to meet various data analysis needs. [29]

Circle-map BED output files [23] provide the coordinates of the genomic regions that originally formed the eccDNA, so with the usage of Bioconductor package **GenomicRanges** [30] and the annotation package **TxDb.Hsapiens.UCSC.hg38.knownGene** [31] which exposes the annotation database

for *Homo Sapiens* genes generated by coordinates from UCSC, we can identify the genes or part of genes with presence in our circles. **org.Hs.eg.db** [33] annotation package was also used to add Gene Symbols to Entrez gene identifiers.

To this list of genes **RITAN** (rapid integration of term annotation and network resources) enrichment package was used [34], to identify the biologic functions of the genes in eccDNA. RITAN offers the possibility to use many enriched terms databases at the same time, but that makes the output confusing and difficult to show in the app. Reactome [35] was the database of choice, which contains human pathways and processes

Unfortunately, making the genomic information output as reactive as the plot, makes the performance of the app slower and more cumbersome, so this part of the app is static. Bad quality circles have been removed for genomic analysis.

4.3.3 “Circle info” tab

By clicking at any point in the circles output plots in “View Results” tab, you access the last part of the app, which has 3 parts:

- Information about the specific eccDNA circle you have clicked on
- Content of different nucleobases in a circular plot
- A table with the genes or part of genes contained in your circle. If there are no matches in the circle,the genomic data will show *“No genes (or parts of genes) found in this circle.”*

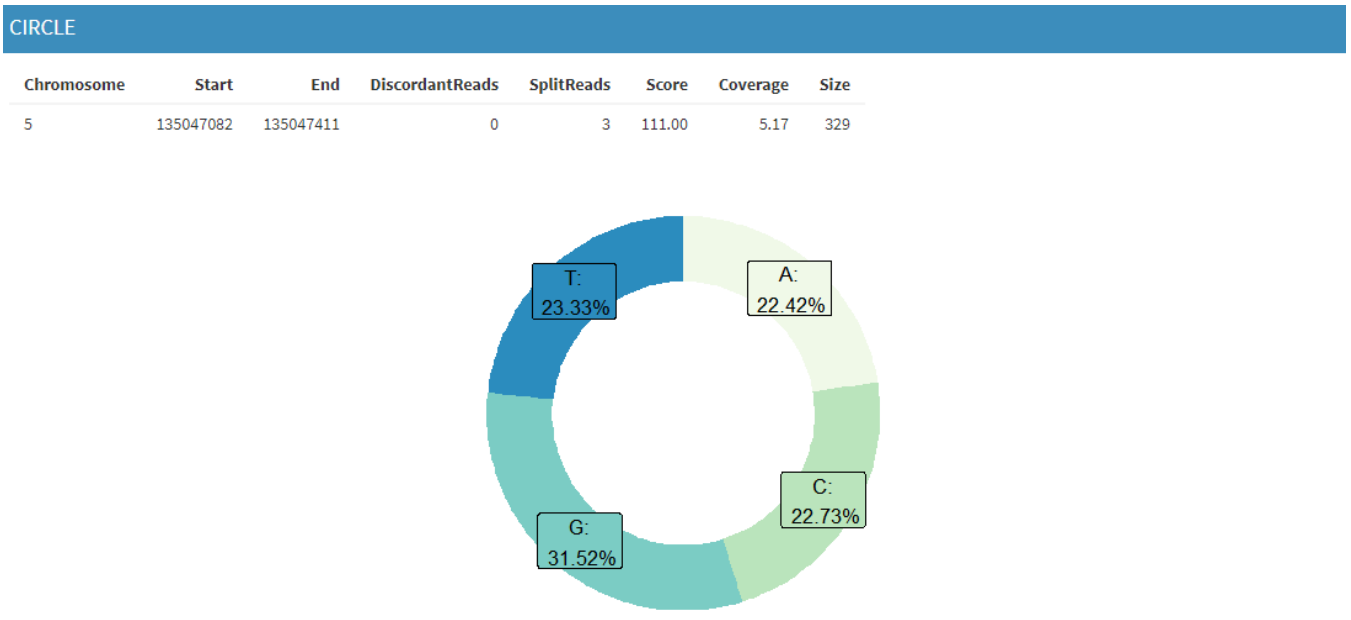


Fig 14. Individual eccDNA output

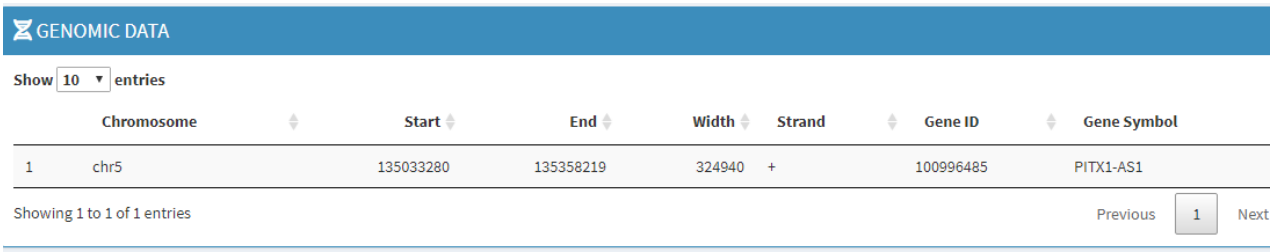


Fig 15. Genes (or parts of genes) contained in individual eccDNA

To achieve this output, first it was necessary to filter the whole dataset with the information provided by the click-in plot event. Then, using the coordinates of that specific instance the different base content data is obtained with the Bioconductor package **Biostrings** [36], for manipulation of biological sequences and the **BSgenome.Hsapiens.UCSC.hg38** annotation package [37], that contains full genome sequences for Homo sapiens (Human) as provided by UCSC (hg38, based on GRCh38.p12) and stored in Biostrings objects.

The circular plot was obtained using polar coordinates, and the process to get the genes contained in individual eccDNA was accomplished in the same manner as explained in the previous section.

Package name	Repository	Use
shiny/shinydashboard	*CRAN*	*Web app dashboard development*
dplyr	*CRAN*	*Data manipulation*
ggplot2	*CRAN*	*Plots based on "The Grammar of Graphics"*
stringr	*CRAN*	*String manipulation*
DT	*CRAN*	*Render data objects in R as HTML tables*
plotly	*CRAN*	*Interactive plots*
GenomicRanges	*Bioconductor (software)*	*Process genomic coordinates*
TxDb.Hsapiens.UCSC.hg38.knownGene	*Bioconductor (annotation)*	*Coordinates for human genes in TxDb*
org.Hs.eg.db	*Bioconductor (annotation)*	*Entrez Gene identifiers and Gene Symbols*
RITAN	*Bioconductor (software)*	*Enriched pathway annotation (reactome)*
Biostrings	*Bioconductor (software)*	*Manipulation of biological sequences*
BSgenome.Hsapiens.UCSC.hg38	*Bioconductor (annotation)*	*Full genome sequences for Homo Sapiens*

Table 2. Packages used in the development of Visual eccDNA

5. Results

The full access for the user to Visual eccDNA is possible in 2 ways: locally or online.

5.1 Local

The R code of the app is hosted in the GitHub repository used for VCS (https://github.com/agarcia18/eccDNA) and it is also provided in the annexes of this report. Example BED files for testing the app are also provided in this repository.

After the repository is downloaded it is possible to call a function to run Visual eccDNA locally on the computer. There is a little script called `install_packages.R` that will download and install the packages if needed, but to run the app, Shiny libraries must be installed. So after setting eccDNA folder as working directory, these are the lines of code to load Visual eccDNA (It is saved in the repository as `run_app.R)`.

```
if("shiny" %in% rownames(installed.packages()) == FALSE)
{install.packages("shiny")}
if("shinydashboard" %in% rownames(installed.packages()) ==
FALSE) {install.packages("shinydashboard")}

shiny::runApp("./visualeccdna_app")
```

In the eccDNA repository, there is also a script to build a Docker image and run it locally. Docker can package an application and its dependencies in a virtual container that can run on any Linux, Windows, or macOS computer. With a Dockerfile, we tell Docker how to build our new image. A Dockerfile is a text file and by default is assumed to be located in the build-context root directory [37] The folder for that purpose is called `docker_local_visualeccdna`.

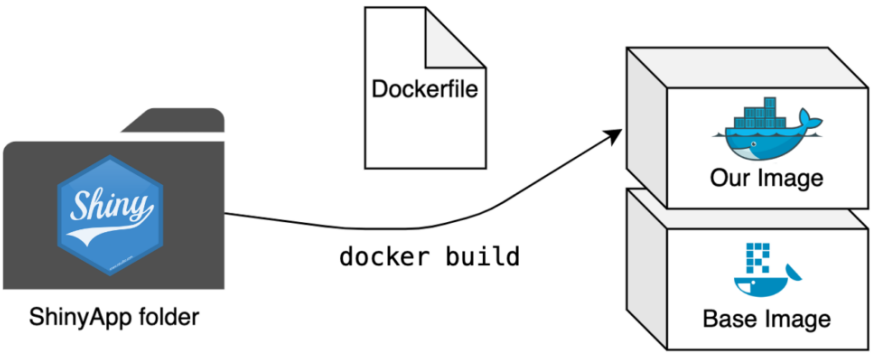


Fig 16. Process for building a Docker image [37]

After Docker is installed in the computer, the following lines must be run in the console of that folder to first build and then to run Visual eccDNA. The Dockerfile code is also in the GitHub repository and in the annexes.

```
# Build image
docker build -t my-shinyapp-image .

# Start a container
docker run -d --rm -p 3838:3838 my-shinyapp-image
```

5.2 Online

Docker containers can also be shared easily online. Professor I.Mallona, kindly deployed Visual eccDNA project in her personal web host, so it is available in http://imlsauftakt.uzh.ch:3845/visualeccdna/.

6. Biological discussion

Visual eccDNA was developed to resolve two main problems: make accessible and easy to understand the output of bioinformatics data for people from different backgrounds and to compare results of different circular mapping pipelines that are being developed. The biological results obtained from the data was not very important, but the way to visualize it.

Anyhow, it is important to confirm that the app is not returning nonse biological results. The raw data was obtained from healthy muscle tissue and from leukocytes [16], so we can find in the Reactome pathway enrichment results in this connection, especially in the files that contain a big number of eccDNA.

For example, in fig.18 processes involved in antigen presentation or adaptive immune system are representative, indicating that the sample is from leukocytes. In contrast, in fig.19 axon guidance, synapsis and neural system pathways count for more, so it is a muscle sample. Each of the plots come from different files.

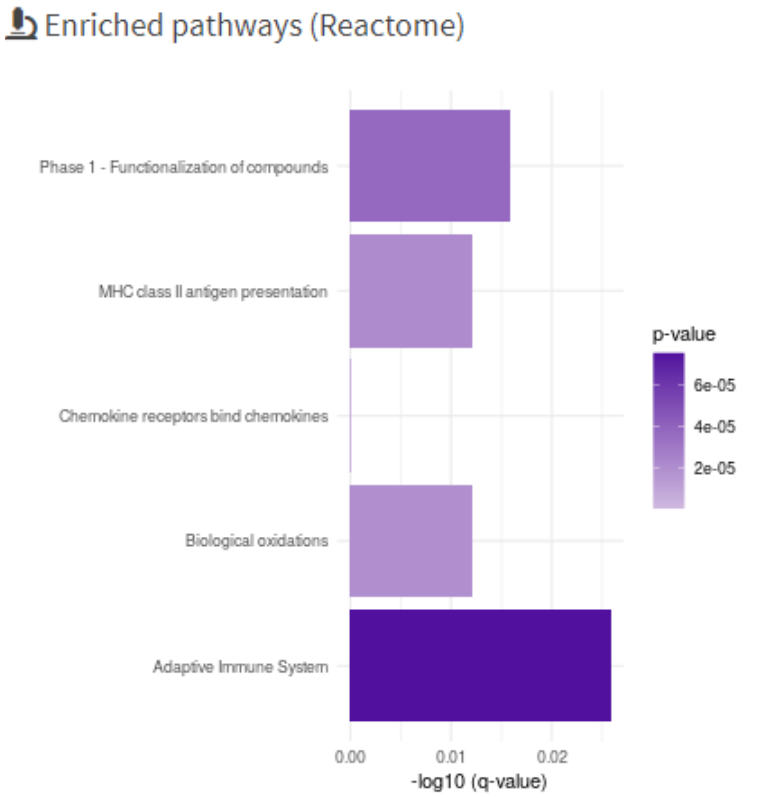


Fig 17. Pathways involved in immune regulation

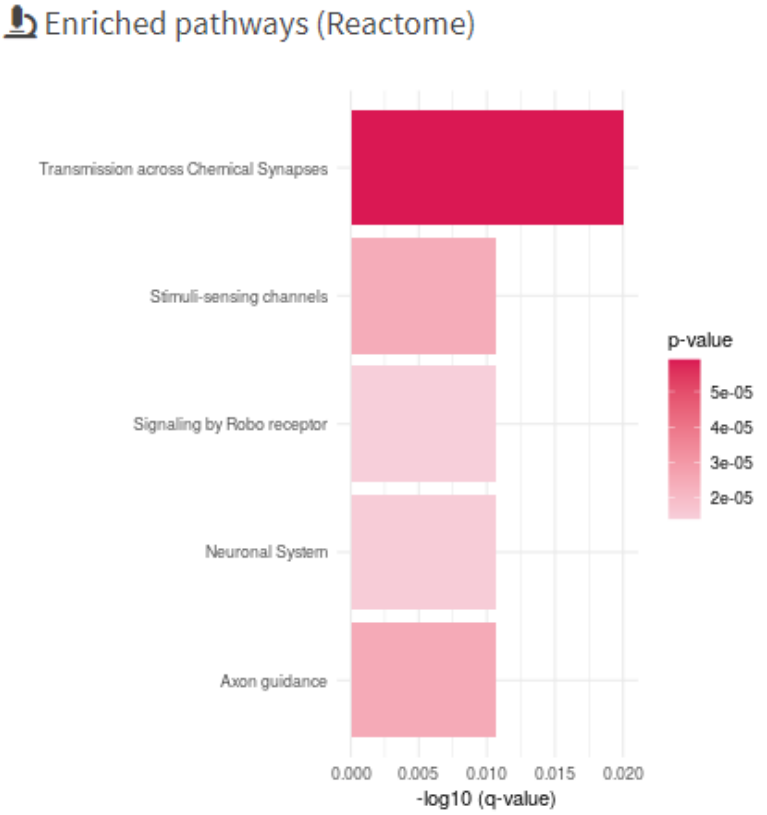


Fig 18. Pathways involved in sinapsis

Discordant reads and split reads are a signal for eccDNA detection in bioinformatic tools so having this in a hovering effect in the dynamic plot for circle selection was important. However, the sequence content in bases for each circle is not mapped back to the real data, but to the reference genome,so if there are mismatches Visual eccDNA will not show them. Especially if there is a significant rise in the CG content, this will increase the stability of the eccDNA and also GC-rich isochores typically include many protein-coding genes within them [38]. As eccDNA replicates uncontrollably and genes in them are overexpressed, it is important to find these kinds of patterns for genomic regulation.

7. Conclusions

7.1 General conclusions and future lines of research

Being involved in this project was an enriching experience to learn more about mechanisms involved in genomic regulation, oncology and of course, bioinformatics. The development of NGS allows the researchers to design experiments like Circle-seq for eccDNA genomic characterization and revive a subject that had reached a technological bottleneck. Understanding the process behind bioinformatic tools for eccDNA detection amplified the RNA-seq experiments subjects covered in the master.

The development of this app in Shiny helped with the usage of R language, exploring new libraries and being autonomous about the decisions of how to achieve the best result for a problem given, the visualization of eccDNA genomic data. It was a totally new subject and very instructive and useful to learn that also amplified the master's knowledge.

Given the time for the collaboration with Biodonostia HRI, comparing results of different circular mapping pipelines that are being developed was not possible. Only Circle-map BED files output were used as input for Visual eccDNA. This is the main tool used now by the European team in collaboration with Araúzo's lab. If new tools are developed in the future, Visual eccDNA can be adapted to fit new output data, but the process for obtaining genomic information will always rely on coordinates.

7.2 Monitoring of the working plan

When the working on the project began, more time was needed in the process of making the layout and overall display of the information of Visual eccDNA, as well as the correct treatment of the files and page interactivity.

This delayed the design of the circle graph and the process of the genomic information, but it was necessary. Shiny dashboard code works differently when you just show the output of the data hosted in your computer or when it is uploaded from third parties.

Due to my inexperience in this field, this was fistly planned for the end of the project. Luckily, taking this into consideration at the beginning, resulted in a more robust app, in contrast to having to adapt the code for the uploaded data by the end of the project.

There was also a problem when deploying the app. The first approach was to use shinyapps.io, which is a very easy to use platform provided by RStudio but Visual eccDNA did not get to work fully in this platform. It processed the BED files correctly and shared the first tab great, but when moving to other tabs it crashed. I tried to solve the problem, but I think it was more related with the reactivity and click in events.

This was solved , because Visual eccDNA, although it is designed to be shared within different working groups,is not mandatory to be working online. It can be shared, built in different computers and used locally, either downloading the app.R script from the GitHub repository or building the Docker image. However, I.Mallona helped in this process and kindly deployed and hosted the app in http://imlsauftakt.uzh.ch:3845/visualeccdna/.

8. Glossary

- eccDNA - Extrachromosomal circular DNA: Genetic material outside of the chromosomes, with transcriptomic capability and lack of regulation.

- Biodonostia HRI: Health Research Institute in San Sebastian associated with the hospital

- Circle-seq: NGS technique for genomic characterization of eccDNA. Involves purifying and amplifying circles for library preparation.

- Circle-map: Bioinformatic tool for eccDNA detection.

- BED (Browser Extensible Data) format: text file format used to store genomic regions as coordinates and associated annotations separated by tabs.

- Circle map output BED file: specific kind of BED file provided by Circle-map.

- VCS - Version Control System: a kind of software that helps the developer manage all the changes that have been made during the development.

- R: programming language oriented to statistics.

- Shiny: R library to design interactive web applications.

- Dashboard: a graphical interface for users to quickly visualize important metrics.

- Tab: different parts (pages) of the dashboard.

- Bioconductor: open code projecto for genomic data processing in R.

- CG content: Percentage of guanine (G) and cytosine (C).

9. References

1. Hoon Kim, Nam-Phuong Nguyen, Kristen Turner, Sihan Wu, Amit D Gujar, Jens Luebeck, Jihe Liu, Viraj Deshpande, Utkrisht Rajkumar, Sandeep Nambur, Samirkumar B Amin, Eunhee Yi, Francesca Menghi, Johannes H Schulte, Anton G Henssen, Howard Y Chang, Christine R Beck, Paul S Mischel, Vineet Bafna, Roel G W Verhaak (Aug 2020) - "Extrachromosomal DNA is associated with oncogene amplification and poor outcome across multiple cancers."

2. Biodonostia Health Research Institute – Computational biology

3. DavidCox, CatherineYuncken, ArthurI.Spriggs, M.C.Path (1965) - "Minute chromatin bodies in malignant tumours of chilminute chromatin bodies in malignant tumours of childhood"

4. Bertelsen AH, et al. (1982) - "Molecular characterization of small polydisperse circular deoxyribonucleic acid from an African green monkey cell line."

5. The curious DNA circles that make treating cancer so hard (2020)

6. Laura W. Dillon,Pankaj Kumar, Yoshiyuki Shibata, Yuh-Hwa Wang, Smaranda Willcox, Jack D. Griffith, Yves Pommier, Shunichi Takeda and Anindya Dutta (2015) - "Production of extrachromosomal microDNAs is linked to mismatch repair pathways and transcriptional activity."

7. Frederic G. Barr, Lauren E. Nauta, Richard J. Davis, Beat W. Schäfer, Lynn M. Nycum and Jaclyn A. Biegel (1996) - "In vivo amplification of the PAX3-FKHR and PAX7-FKHR fusion genes in alveolar rhabdomyosarcoma".

8. Ofer Shoshani, Simon F. Brunner, Rona Yaeger, Peter Ly, Yael Nechemia-Arbely, Dong Hyun Kim, Rongxin Fang, Guillaume A. Castillon, Miao Yu, Julia S. Z. Li, Ying Sun, Mark H. Ellisman, Bing Ren, Peter J. Campbell & Don W. Cleveland (2020) - "Chromothripsis drives the evolution of gene amplification in cancer"

9. Vukovic B., Beheshti B., Park P, Lim G., Bayani J., Zielenska M.b, Squire J.A. (2007) - "Correlating breakage-fusion-bridge events with the overall chromosomal instability and in vitro karyotype evolution in prostate cancer".

10. Clelia Tiziana Storlazzi , Angelo Lonoce, Maria C Guastadisegni, Domenico Trombetta, Pietro D'Addabbo, Giulia Daniele, Alberto L'Abbate, Gemma Macchia, Cecilia Surace, Klaas Kok, Reinhard Ullmann, Stefania Purgato, Orazio Palumbo, Massimo Carella, Peter F Ambros, Mariano Rocchi (2010) - "Gene amplification as double minutes or homogeneously staining regions in solid tumors: origin and structure".

11. Wu S, Turner KM, Nguyen N, Raviram R, Erb M, Santini J, et al. (2019) - "Circular ecDNA promotes accessible chromatin and high oncogene expression."

12. Pankaj Kumar, Shashi Kiran, Shekhar Saha, Zhangli Su, Teressa Paulsen, Ajay

Chatrath, Yoshiyuki Shibata, Etsuko Shibata, Anindya Dutta (2020) - "ATAC-seq identifies thousands of extrachromosomal circular DNA in cancer and cell lines."

13. Man Wang, Xinzhe Chen, Fei Yu, Han Ding, Yuan Zhang and Kun Wang (2021) - "Extrachromosomal Circular DNAs: Origin, formation and emerging function in Cancer."

14. Mengdi Cai, Huishu Zhang, Liqing Hou, Wei Gao, Ying Song, Xiaobo Cui, Chunxiang Li, Rongwei Guan, Jinfa Ma, Xu Wang, Yue Han, Yafan Lv, Feng Chen, Ping Wang, Xiangning Meng, Songbin Fu (2019) - "Inhibiting homologous recombination decreases extrachromosomal amplification but has no effect on intrachromosomal amplification in methotrexate-resistant colon cancer cells."

15. Zhi-Xiong Cai, Geng Chen, Yong-Yi Zeng, Xiu-Qing Dong, Min-Jie Lin, Xin-Hui Huang, Da Zhang, Xiao-Long Liu, Jing-Feng Liu (2017) - "Circulating tumor DNA profiling reveals clonal evolution and real-time disease progression in advanced hepatocellular carcinoma."

16. Henrik Devitt Møller, Marghoob Mohiyuddin, Iñigo Prada-Luengo, M Reza Sailani, Jens Frey Halling, Peter Plomgaard, Lasse Maretty, Anders Johannes Hansen, Michael P Snyder, Henriette Pilegaard, Hugo Y K Lam, Birgitte Regenberg (2018) - "Circular DNA elements of chromosomal origin are common in healthy human somatic tissue."

17. Sarah T. K. Sin, Peiyong Jiang, Jiaen Deng, Lu Ji, Suk Hang Cheng, Anindya Dutta, Tak Y. Leung, K. C. Allen Chan, Rossa W. K. Chiu, and View ORCID ProfileY. M. Dennis Lo (2019) - "Identification and characterization of extrachromosomal circular DNA in maternal plasma."

18. Xiao-Ou Zhang, Rui Dong, Yang Zhang, Jia-Lin Zhang, Zheng Luo, Jun Zhang, Ling-Ling Chen and Li Yang (2016) - " Diverse alternative back-splicing and alternative splicing landscape of circular RNAs."

19. Iñigo Prada-Luengo, Anders Krogh, Lasse Maretty & Birgitte Regenberg (2019) - "Sensitive detection of circular DNAs at single-nucleotide resolution using guided realignment of partially aligned reads."

20. Git and GitHub: What's the difference? - https://blog.devmountain.com/git-vs-github-whats-the-difference/

21. R Shiny- https://shiny.rstudio.com/

22. R Shiny Dashboard - https://rstudio.github.io/shinydashboard/index.html

23. Circle-map output files wiki- https://github.com/iprada/Circle-Map/wiki/Circle-Map-Realign-output-files

24. R Dplyr - https://dplyr.tidyverse.org/

25. The Grammar of graphics - https://cfss.uchicago.edu/notes/grammar-of-graphics/

26. R Stringr - https://cran.r-project.org/web/packages/stringr/vignettes/stringr.html

27. DT: an R interface to the DataTables library - https://rstudio.github.io/DT/

28. Getting Started with Plotly - https://plotly.com/r/getting-started/

29. Gentleman, Robert C.; Carey, Vincent J.; Bates, Douglas M.; Bolstad, Ben; Dettling, Marcel; Dudoit, Sandrine; Ellis, Byron; Gautier, Laurent; Ge, Yongchao; Gentry, Jeff; Hornik, Kurt; Hothorn, Torsten; Huber, Wolfgang; Iacus, Stefano; Irizarry, Rafael; Leisch, Friedrich; Li, Cheng; Maechler, Martin; Rossini, Anthony J.; Sawitzki, Gunther; Smith, Colin; Smyth, Gordon; Tierney, Luke; Yang, Jean Y. H.; Zhang, Jianhua (2004) - "Bioconductor: open software development for computational biology and bioinformatics".

30. Michael Lawrence ,Wolfgang Huber,Hervé Pagès,Patrick Aboyoun,Marc Carlson,Robert Gentleman,Martin T. Morgan,Vincent J. Carey (2013) - "Software for Computing and Annotating Genomic Ranges"

31. Bioconductor TxDb.Hsapiens.UCSC.hg38.knownGene annotation package - https://bioconductor.org/packages/release/data/annotation/html/TxDb.Hsapiens.UCSC.hg38.knownGene.html

32. Bioconductor org.Hs.eg.db annotation package -https://bioconductor.org/packages/release/data/annotation/html/org.Hs.eg.db.html

33. Michael T. Zimmermann, Brian Kabat, Diane E. Grill, Richard B. Kennedy, and Gregory A. Poland (2019) - "RITAN: rapid integration of term annotation and network resources"

34. David Croft, Gavin O'Kelly, Guanming Wu, Robin Haw, Marc Gillespie,Lisa Matthews, Michael Caudy,Phani Garapati, Gopal Gopinath,Bijay Jassal, Steven Jupe, Irina Kalatskaya, Shahana Mahajan,Bruce May, Nelson Ndegwa, Esther Schmidt, Veronica Shamovsky,Christina Yung, Ewan Birney, Henning Hermjakob, Peter D'Eustachio, and Lincoln Stein (2010) - "Reactome: a database of reactions, pathways and biological processes"

35. Pagès H, Aboyoun P, Gentleman R, DebRoy S (2020) - "Biostrings: Efficient manipulation of biological strings"

36. *Bsgenome.Hsapiens.UCSC.hg19* - Full genome sequences for Homo sapiens (UCSC version hg19, based on GRCh37.p13) https://bioconductor.org/packages/release/data/annotation/html/BSgenome.Hsapiens.UCSC.hg19.html

37. How to Dockerize Shiny apps - https://www.statworx.com/de/blog/how-to-dockerize-shinyapps/

38. BrahimAïssani GiorgioBernardi (1991) - "CpG islands, genes and isochores in the genomes of vertebrates"

10. Annexes

run_app.R

```
if("shiny" %in% rownames(installed.packages()) == FALSE)
{install.packages("shiny")}
if("shinydashboard" %in% rownames(installed.packages()) ==
FALSE) {install.packages("shinydashboard")}

shiny::runApp("./visualeccdna_app")
```

install_packages.R

```
using<-function(...) {
  libs<-unlist(list(...))
  req<-unlist(lapply(libs,require,character.only=TRUE))
  need<-libs[req==FALSE]
  n<-length(need)
  if(n>0){
        libsmsg<-if(n>2) paste(paste(need[1:(n-1)],collapse=",
"),",",sep="") else need[1]
    print(libsmsg)
    if(n>1){
      install.packages(need)
      lapply(need,require,character.only=TRUE)
    }
  }
}

bioc_using<-function(...) {
  libs<-unlist(list(...))
  req<-unlist(lapply(libs,require,character.only=TRUE))
  need<-libs[req==FALSE]
  n<-length(need)
  if(n>0){
        libsmsg<-if(n>2) paste(paste(need[1:(n-1)],collapse=",
"),",",sep="") else need[1]
    print(libsmsg)
    if(n>1){
      libsmsg<-paste(libsmsg," and ", need[n],sep="")
    }
      BiocManager::install(need)
      lapply(need,require,character.only=TRUE)
    }
  }
```

```
using("shinydashboard","dplyr","ggplot2","stringr","plotly","D
T")
install.packages("BiocManager",version=3.13)
bioc_using("TxDb.Hsapiens.UCSC.hg38.knownGene","GenomicRanges"
,"org.Hs.eg.db","RITANdata","RITAN","Biostrings","BSgenome.Hsa
piens.UCSC.hg38")
```

app.R

```
# Load required packages
source("./install_packages.R")

# SIZE FILTERING
circ_size <- 5000
range_small <- c(250,2500)
range_big <- c(10000,1000000)

################################UI#########################

ui <- dashboardPage(

    dashboardHeader(title = "Visual eccDNA"),

    dashboardSidebar(
        sidebarMenu(
            id="tabs",
                                                menuItem("Get
started",tabName="about",icon=icon("cloud-upload")),
                              menuItem("View  results",  tabName
="results",icon=icon("bar-chart"), startExpanded = TRUE,
                            menuSubItem(paste0("Circles  under  ",
circ_size,"bp"), tabName = "smallcirc"),
                             menuSubItem(paste0("Circles  over  ",
circ_size,"bp"), tabName = "bigcirc")),
                                                menuItem("Circle
Info",tabName="circle",icon=icon("circle-o")),
                                    menuItem("GitHub",tabName   =
"github",icon=icon("github"))
        )
        ),

    dashboardBody(
        tabItems(
            tabItem(
                tabName = "about",
                                      fluidRow(HTML('<center><img
src="https://i.ibb.co/7tj8wsG/logo.png"></center>'),
                              br(),br(),
```

```
                        box(width = 12, solidHeader = TRUE,
                         "Wellcome to Visual eccDNA. This is an
app for dynamic data visualization of Extrachromosomal
Circular DNA (eccDNA) output files. Start selecting a BED file
and this page will show an overview of the results. For better
understanding of the output, circles will be separated by
size. You can press the buttons View Results, and you will be
able to see further information about the DNA circles
contained in your file."
                    )),
                    fluidRow(
                                box(title= "Upload a Circle BED
file   with   output:",fileInput("bedfile","Choose   file:"),
status="primary"),
                            uiOutput("total")
                            ),
                    fluidRow(uiOutput("quality_gg"),
                             uiOutput("click_small"),
                             uiOutput("click_big")
                             ),
                    fluidRow(uiOutput("table"))
            ),
            tabItem(
                tabName = "smallcirc",
                    fluidRow(
                        uiOutput("plot_results_small"),
                             uiOutput("filters_small"),
                             uiOutput("backbuttonsmall"),

                        ),
                    fluidRow(uiOutput("table_small"),
                             uiOutput("pathway_small"))
                    ),
            tabItem(
                tabName = "bigcirc",
                fluidRow(uiOutput("plot_results_big"),
                         uiOutput("filters_big"),
                         uiOutput("backbuttonbig")
                         ),
                fluidRow(uiOutput("table_big"),
                         uiOutput("pathway_big"))
            ),
            tabItem(
                tabName="circle",
                    fluidRow(uiOutput("selected_circle"))
                ),
```

```
            tabItem(
                tabName="github",
                   "You can follow the development and version
control of this app in:",br(),

tags$a(href="https://github.com/agarcia18/eccDNA","https://git
hub.com/agarcia18/eccDNA")
            )
        )
    )
)

############################SERVER##############################
##
server <- function(input, output,session){
    # Change options to process bigger files (30MB)
    options(shiny.maxRequestSize=30*1024^2)



    # Import data from file and add labels
    dataframe<-reactive({
        if (is.null(input$bedfile))
            return(NULL)
             circ <-read.table(input$bedfile$datapath,header =
FALSE, sep="\t",stringsAsFactors=FALSE)
                                                names(circ)     <-
c("chrom","start","end","discordant_reads","split_reads","scor
e","coverage_mean","coverage_sd","coverage_start",
"coverage_end","coverage_cont")

        # Add circle id
             circ <- circ %>% mutate(circle_id = 1:n()) %>%
dplyr::select(circle_id, everything())

          # Add quality levels (score < 10 = Bad, score < 50 =
Low, score < 200 = Medium, score > 200 = Good)
                                                circ$quality     <-
cut(circ$score,breaks=c(-Inf,10,50,200,Inf),labels=
c("Bad","Low", "Medium", "Good"),right = FALSE)

        # Add size of circle (number of bp)
        circ$size_bp <- circ$end - circ$start

        circ })


##############       "Get       Started"       tab       server
```

```
functions##############

    # TOTAL CIRCLES VALUEBOX
    output$total <- renderUI({
        req(input$bedfile)
          valueBox(value=dataframe() %>% count(),tags$b("TOTAL
eccDNA CIRCLES"), color="navy",width=6,
                       icon = tags$i(class = "far fa-circle",
style="color:white"))
    })


    # VIEW RESULTS BUTTONS - SMALL & BIG CIRCLES

    # Small circles % and click button
    output$click_small<-renderUI({
        req(input$bedfile)
         perc<-as.data.frame(dataframe() %>% group_by(size_bp <
circ_size)     %>%     count()     %>%     mutate(pct_tot     =
n/nrow(dataframe())*100))
        perc_small <- perc[2,3]

        box(background = "maroon",height=150,width=3,
            h4("Small eccDNA circles:",align="center"),

h1(paste0(round(perc_small,2),"%"),align="center"),
                           actionButton("click_smallcirc","View
Results"),align="center")
    })

    # Click event - Change to the small circles plot tab
    observeEvent(input$click_smallcirc, {
        newtab <- switch(input$tabs,
                         "about" = "smallcirc")

        updateTabItems(session, "tabs", newtab)
    })


   # Big circles % and click button
    output$click_big<-renderUI({
        req(input$bedfile)
         perc<-as.data.frame(dataframe() %>% group_by(size_bp <
circ_size)     %>%     count()     %>%     mutate(pct_tot     =
n/nrow(dataframe())*100))
        perc_big <- perc[1,3]
```

```
        box(background = "purple",height=150,width=3,
            h4("Big eccDNA circles:",align="center"),
            h1(paste0(round(perc_big,2),"%"),align="center"),
                                actionButton("click_bigcirc","View
Results"),align="center")
    })
    # Click event - Change to the big circles plot tab
    observeEvent(input$click_bigcirc, {
        newtab2 <- switch(input$tabs,
                          "about" = "bigcirc")

        updateTabItems(session, "tabs", newtab2)
    })

    # QUALITY PLOT
    output$quality_gg<-renderUI({

        # Read data and add labels
        req(input$bedfile)
        if (is.null(input$bedfile))
            return(NULL)
                                box(title="Quality    in    eccDNA
circles",height=300,width=6,status="primary",
            renderPlot(
            ggplot(dataframe(), aes(x = quality)) +
                                                geom_bar(aes(y   =
(..count..)/sum(..count..),fill=quality),width=0.4,alpha=0.8,p
osition = position_stack(reverse = TRUE))+

scale_fill_manual(values=c("#FF5733","#FFC300","#DAF7A6","#6EB
C63"))+
                scale_y_continuous(labels=scales::percent)+
                xlab("")+ylab("")+coord_flip()+
                theme_minimal()+
                guides(fill="none")
            ,height=200),
                p(em("Quality is calculated from score: Bad (<
10),  Low  (10-50),  Medium  (50-200),  Good  (>  200).",style  =
"font-size:12px;")))
    })

    # DATA TABLE
    output$table <- renderUI({
        req(input$bedfile)
        box(width = NULL, solidHeader = TRUE,
            DT::renderDataTable({
            DT::datatable(dataframe(),options = list(
```

```
                searching = FALSE,
                pageLength =8 ,
                lengthMenu = c(5, 10, 15, 20),
                scrollX=TRUE,
                          columnDefs = list(list(className =
'dt-center', targets = 0:4))
            ))
        })
        )
    })

#####  "View  Results"  tab  server  functions  -  Small  circles
######
    # PLOT OUTPUT - SMALL CIRCLES

    # Import data for plot and make it reactive
    data_circ_small <- reactive({
        if (is.null(input$bedfile))
            return(NULL)
             circ <-read.table(input$bedfile$datapath,header =
FALSE, sep="\t",stringsAsFactors=FALSE)

                                              names(circ)     <-
c("chrom","start","end","discordant_reads","split_reads","scor
e","coverage_mean","coverage_sd","coverage_start",
"coverage_end","coverage_cont")

        # Set chromosome as factor and fix random outputs
        circ$chrom <- substr(circ$chrom, start = 1, stop = 5)
        circ$chrom<-str_remove(circ$chrom,"_")
        circ$chrom<-str_remove(circ$chrom,"chr")
                 circ$chrom  <-  factor(circ$chrom,  levels  =
c("1","2","3","4","5","6","7","8","9","10","11","12","13","14"
,"15","16","17","18","19","20","21","22","Un","M","X","Y"))



          # Add quality levels (score < 10 = Bad, score < 50 =
Low, score < 200 = Medium, score > 200 = Good)
                                             circ$quality     <-
cut(circ$score,breaks=c(-Inf,10,50,200,Inf),labels=
c("Bad","Low", "Medium", "Good"),right = FALSE)

        # Add circle id
             circ  <-  circ  %>%  mutate(circle_id  =  1:n())  %>%
dplyr::select(circle_id, everything())

        # Add size of circle (number of bp)
```

```
        circ$size_bp <- circ$end - circ$start

           # Remove bad coverage circles and wrong discordant
reads outputs
                 circ  <-  filter(circ,coverage_cont  <  0.5  &
discordant_reads < size_bp)

        # Incorporate filters input (size and quality)
              data_circ_small<-  filter(circ,  between(size_bp,
input$size_small[1],    input$size_small[2]),    quality    ==
input$quality_small)
    })

    # Plot for small circles
    output$plot_results_small <- renderUI({
        req(input$bedfile)
         box(title=span(icon("circle-o"), paste0("Circles under
",   circ_size,"bp")),width=9,   height   =   460,solidHeader   =
TRUE,background="maroon",
            renderPlotly({
                fig <- data_circ_small() %>%
                    plot_ly(type = 'scatter',
                            source="smallcircleSource",
                            mode = 'markers',
                            marker = list(
                                size = 8),
                            color = ~chrom,
                            x = ~chrom,
                            y = ~size_bp,
                            text = ~discordant_reads,
                            customdata=~split_reads,
                                          hovertemplate  =
paste("<b>Discordant Reads: %{text}<br>",
                                                         "Split
Reads: %{customdata}<br>",
                                                         "Size:
%{y:.0} bp <br>"),
                            showlegend = FALSE)
                fig <- fig  %>% layout(xaxis = list(
                      title = "Chromosome of origin"), yaxis =
list(title="Size (number of base pairs)"))
                fig
            })
        )
    })
```

```
    # BOX OF FILTERS FOR THE PLOT - SMALL CIRCLES
    output$filters_small<-renderUI({
        req(input$bedfile)
                    box(title=span(icon("filter"),"Filters  (for
plot)"),width=3,
            sliderInput(inputId = "size_small",
                        "By size (number of base pairs):",
                        min = 10,
                        max = circ_size,
                        value = range_small),br(),
            checkboxGroupInput(inputId = "quality_small",
                               "By quality:",
                                                  choices  =
c("Bad","Low","Medium","Good"),
                               selected= c("Medium","Good")),
        )
    })

    # "BACK TO GET STARTED" BUTTON

    # Render the button
    output$backbuttonsmall<-renderUI({
        req(input$bedfile)
        box(width=2,
              actionButton("click_backsmallcirc","Back  to  Get
Started"),align="center")
    })

    # Click event - Change tab
    observeEvent(input$click_backsmallcirc, {
        back1 <- switch(input$tabs,
                        "smallcirc" = "about")

        updateTabItems(session, "tabs", back1)
    })

    # TABLE OF GENES IN SMALL CIRCLES
    output$table_small <- renderUI({
        req(input$bedfile)

        # Import the data
             circ  <-read.table(input$bedfile$datapath,header  =
FALSE, sep="\t",stringsAsFactors=FALSE)
                                              names(circ)     <-
c("chrom","start","end","discordant_reads","split_reads","scor
e","coverage_mean","coverage_sd","coverage_start",
```

```
"coverage_end","coverage_cont")
        circ$size_bp <- circ$end - circ$start
                                                circ$quality     <-
cut(circ$score,breaks=c(-Inf,10,50,200,Inf),labels=
c("Bad","Low", "Medium", "Good"),right = FALSE)

        # Process it to get gene list
            coords<- circ %>% filter (size_bp < circ_size &
quality   !="Bad")   %>%   dplyr::select(chrom,start,end)   %>%
makeGRangesFromDataFrame
        genes <-genes(TxDb.Hsapiens.UCSC.hg38.knownGene)
                                                 genes_df      <-
as.data.frame(subsetByOverlaps(genes,coords))
        entrezid <- genes_df$gene_id
        hs <- org.Hs.eg.db
        genesymbol_df<- AnnotationDbi::select(hs,
                                               keys = entrezid,
                                                       columns =
c("ENTREZID", "SYMBOL"),
                                                       keytype =
"ENTREZID")
                  names(genesymbol_df)[names(genesymbol_df)  ==
"ENTREZID"] <- "gene_id"
        gene_list_small<-left_join(genes_df,genesymbol_df)

names(gene_list_small)<-c("Chromosome","Start","End","Width","
Strand","Gene ID", "Gene Symbol")

        # Render the data table
         box(title=span(icon("fal fa-dna"),"Genes in circles"),
width = 7, solidHeader = TRUE,
            DT::renderDataTable({
                                DT::datatable(gene_list_small[,
names(gene_list_small) != "Gene ID"],options = list(
                    autoWidth = TRUE,
                    searching = FALSE,
                    pageLength =6 ,
                    lengthMenu = c(5, 10, 15, 20),
                    scrollX=TRUE,
                            columnDefs = list(list(className =
'dt-center', targets = 0:5))
                  )) %>% DT::formatStyle(columns = colnames(.),
fontSize = '8pt')
            })
        )
        })

    # PLOT - PATHWAY ENRICHMENT - SMALL CIRCLES
```

```
    output$pathway_small<-renderUI({
        req(input$bedfile)

        # Import the data
            circ <-read.table(input$bedfile$datapath,header =
FALSE, sep="\t",stringsAsFactors=FALSE)
                                              names(circ)     <-
c("chrom","start","end","discordant_reads","split_reads","scor
e","coverage_mean","coverage_sd","coverage_start",
"coverage_end","coverage_cont")
        circ$size_bp <- circ$end - circ$start
                                            circ$quality     <-
cut(circ$score,breaks=c(-Inf,10,50,200,Inf),labels=
c("Bad","Low", "Medium", "Good"),right = FALSE)

        # Get genes list
            coords<- circ %>% filter (size_bp < circ_size &
quality!="Bad")    %>%    dplyr::select(chrom,start,end)    %>%
makeGRangesFromDataFrame
        genes <-genes(TxDb.Hsapiens.UCSC.hg38.knownGene)
                                               genes_df      <-
as.data.frame(subsetByOverlaps(genes,coords))
        entrezid <- genes_df$gene_id
        hs <- org.Hs.eg.db
        genesymbol_df<- AnnotationDbi::select(hs,
                                             keys = entrezid,
                                                    columns =
c("ENTREZID", "SYMBOL"),
                                                    keytype =
"ENTREZID")
                  names(genesymbol_df)[names(genesymbol_df)  ==
"ENTREZID"] <- "gene_id"
        gene_list_small<-left_join(genes_df,genesymbol_df)



        # Enrichment to find pathways
                                            enrich_small     <-
term_enrichment(gene_list_small$SYMBOL,       resources       =
"ReactomePathways", all_symbols = cached_coding_genes)


        # Fix name of pathways for the plot
         enrich_small$name <-sapply(strsplit(enrich_small$name,
split='.', fixed=TRUE), function(x) (x[2]))


           box(title=span(icon("microscope"),"Enriched pathways
(Reactome)"),width=5,

renderPlot(ggplot(enrich_small[1:5,],aes(x=q,y=name,fill=p))+
```

```
                            geom_col()+

scale_fill_gradient(low="#F7CFDA",high="#DA1853")+
                                                    xlab("-log10
(q-value)")+ylab("")+labs(fill = "p-value")+
                            theme_minimal()

                )
            )
    })

###### "View Results" tab server functions - Big circles
#######
    # PLOT OUTPUT - BIG CIRCLES

    # Import data and make it reactive
    data_circ_big <- reactive({
        if (is.null(input$bedfile))
            return(NULL)
             circ <-read.table(input$bedfile$datapath,header =
FALSE, sep="\t",stringsAsFactors=FALSE)

                                              names(circ)     <-
c("chrom","start","end","discordant_reads","split_reads","scor
e","coverage_mean","coverage_sd","coverage_start",
"coverage_end","coverage_cont")

        # Set chromosome as factor and fix random outputs
        circ$chrom <- substr(circ$chrom, start = 1, stop = 5)
        circ$chrom<-str_remove(circ$chrom,"_")
        circ$chrom<-str_remove(circ$chrom,"chr")
                 circ$chrom  <-  factor(circ$chrom,  levels  =
c("1","2","3","4","5","6","7","8","9","10","11","12","13","14"
,"15","16","17","18","19","20","21","22","Un","M","X","Y"))



          # Add quality levels (score < 10 = Bad, score < 50 =
Low, score < 200 = Medium, score > 200 = Good)
                                             circ$quality     <-
cut(circ$score,breaks=c(-Inf,10,50,200,Inf),labels=
c("Bad","Low", "Medium", "Good"),right = FALSE)

        # Add circle id
             circ <- circ %>% mutate(circle_id = 1:n()) %>%
dplyr::select(circle_id, everything())

        # Add size of circle (number of bp)
```

```
        circ$size_bp <- circ$end - circ$start

           # Remove bad coverage circles and wrong discordant
reads outputs
                 circ  <-  filter(circ,coverage_cont  <  0.5  &
discordant_reads < size_bp)

        # Incorporate filters input (size and quality)
                data_circ_big<-  filter(circ,  between(size_bp,
input$size_big[1],      input$size_big[2]),      quality      ==
input$quality_big)
    })

    # Plot of big circles
    output$plot_results_big <- renderUI({
        req(input$bedfile)

         box(title=span(icon("circle-o"), paste0( "Circles over
",  circ_size,"bp")),width=9,  height  =  480,  background  =
"purple",solidHeader = TRUE,

            renderPlotly({
                fig <- data_circ_big() %>%
                    plot_ly(type = 'scatter',
                            source="bigcircleSource",
                            mode = 'markers',
                            marker = list(
                                size = 22),
                            color = ~chrom,
                            x = ~chrom,
                            y = ~size_bp,
                            text = ~discordant_reads,
                            customdata=~split_reads,
                                          hovertemplate  =
paste("<b>Discordant Reads: %{text}<br>",
                                                         "Split
Reads: %{customdata}<br>",
                                                         "Size:
%{y:.0} bp <br>"),
                            showlegend = FALSE
                    )
                fig <- fig  %>% layout(xaxis = list(
                      title = "Chromosome of origin"), yaxis =
list(title="Size (number of base pairs)"))

                fig
            })
```

```
        )
    })


    # BOX OF FILTERS FOR THE PLOT - BIG CIRCLES
    output$filters_big<-renderUI({
        req(input$bedfile)
                box(title=span(icon("filter"),  "Filters  (for
plot)"),width=3,
            sliderInput(inputId = "size_big",
                        "By size (number of base pairs):",
                        min = circ_size,
                        max = 2000000,
                        value = range_big),
            checkboxGroupInput(inputId = "quality_big",
                               "By quality:",
                                                  choices  =
c("Bad","Low","Medium","Good"),
                               selected= c("Medium","Good")),
        )
    })

    # "BACK TO GET STARTED" BUTTON

    #Render the button
    output$backbuttonbig<-renderUI({
        req(input$bedfile)
        box(width=2,
                 actionButton("click_backbigcirc","Back to Get
Started"),align="center")
    })

    # Click event - Back to "Get Started"
    observeEvent(input$click_backbigcirc, {
        back2 <- switch(input$tabs,
                        "bigcirc" = "about")

        updateTabItems(session, "tabs", back2)
    })

    # TABLE OF GENES IN BIG CIRCLES
    output$table_big <- renderUI({
        req(input$bedfile)

        # Import the data
             circ  <-read.table(input$bedfile$datapath,header  =
```

```
FALSE, sep="\t",stringsAsFactors=FALSE)
                                                names(circ)      <-
c("chrom","start","end","discordant_reads","split_reads","scor
e","coverage_mean","coverage_sd","coverage_start",
"coverage_end","coverage_cont")
        circ$size_bp <- circ$end - circ$start
                                               circ$quality      <-
cut(circ$score,breaks=c(-Inf,10,50,200,Inf),labels=
c("Bad","Low", "Medium", "Good"),right = FALSE)


        # Process it to get gene list
            coords<- circ %>% filter (size_bp > circ_size &
quality!="Bad")    %>%    dplyr::select(chrom,start,end)    %>%
makeGRangesFromDataFrame
        genes <-genes(TxDb.Hsapiens.UCSC.hg38.knownGene)
                                                 genes_df      <-
as.data.frame(subsetByOverlaps(genes,coords))
        entrezid <- genes_df$gene_id
        hs <- org.Hs.eg.db
        genesymbol_df<- AnnotationDbi::select(hs,
                                              keys = entrezid,
                                                      columns =
c("ENTREZID", "SYMBOL"),
                                                      keytype =
"ENTREZID")
                  names(genesymbol_df)[names(genesymbol_df)  ==
"ENTREZID"] <- "gene_id"
        gene_list_big<-left_join(genes_df,genesymbol_df)

names(gene_list_big)<-c("Chromosome","Start","End","Width","St
rand","Gene ID", "Gene Symbol")


        # Render the data table
         box(title=span(icon("fal fa-dna"),"Genes in circles"),
width = 7, solidHeader = TRUE,
            DT::renderDataTable({
                                    DT::datatable(gene_list_big[,
names(gene_list_big) != "Gene ID"],options = list(
                    autoWidth = TRUE,
                    searching = FALSE,
                    pageLength =6 ,
                    lengthMenu = c(5, 10, 15, 20),
                    scrollX=TRUE,
                            columnDefs = list(list(className =
'dt-center', targets = 0:4))
                ))
            })
        )
```

```
    })

    # PLOT - PATHWAY ENRICHMENT
    output$pathway_big<-renderUI({
        req(input$bedfile)

        # Import the data
            circ <-read.table(input$bedfile$datapath,header =
FALSE, sep="\t",stringsAsFactors=FALSE)
                                                names(circ)     <-
c("chrom","start","end","discordant_reads","split_reads","scor
e","coverage_mean","coverage_sd","coverage_start",
"coverage_end","coverage_cont")
        circ$size_bp <- circ$end - circ$start
                                               circ$quality     <-
cut(circ$score,breaks=c(-Inf,10,50,200,Inf),labels=
c("Bad","Low", "Medium", "Good"),right = FALSE)

        # Get genes list
            coords<- circ %>% filter (size_bp > circ_size &
quality!="Bad")    %>%    dplyr::select(chrom,start,end)    %>%
makeGRangesFromDataFrame
        genes <-genes(TxDb.Hsapiens.UCSC.hg38.knownGene)
                                                 genes_df      <-
as.data.frame(subsetByOverlaps(genes,coords))
        entrezid <- genes_df$gene_id
        hs <- org.Hs.eg.db
        genesymbol_df<- AnnotationDbi::select(hs,
                                              keys = entrezid,
                                                      columns =
c("ENTREZID", "SYMBOL"),
                                                      keytype =
"ENTREZID")
                  names(genesymbol_df)[names(genesymbol_df)  ==
"ENTREZID"] <- "gene_id"
        gene_list_big<-left_join(genes_df,genesymbol_df)



        # Enrichment to find pathways
           enrich_big <- term_enrichment(gene_list_big$SYMBOL,
resources     =     "ReactomePathways",     all_symbols     =
cached_coding_genes)

        # Fix name of pathways for the plot
            enrich_big$name <-sapply(strsplit(enrich_big$name,
split='.', fixed=TRUE), function(x) (x[2]))

          box(title=span(icon("microscope"),"Enriched pathways
```

```
(Reactome)"),width=5,

renderPlot(ggplot(enrich_big[1:5,],aes(x=q,y=name,fill=p))+
                           geom_col()+

scale_fill_gradient(low="#CEB9DF",high="#51119E")+
                                                    xlab("-log10
(q-value)")+ylab("")+labs(fill = "p-value")+
                           theme_minimal()

            )
        )
    })


#####################################  "Circle info" tab server
functions ####################################

# As we have two plots of output (small & big circles), this
page will be reactive to both of them

    ## SOURCE: SMALL CIRCLES PLOT
     # Change tab and render info when clicking in the plot of
small circles

observeEvent(event_data("plotly_click",source="smallcircleSour
ce"), {
        circletab1 <- switch(input$tabs,
                             "smallcirc" = "circle")

        updateTabItems(session, "tabs", circletab1)
    })


observeEvent(event_data("plotly_click",source="smallcircleSour
ce"), {
        output$selected_circle<- renderUI({
            fluidRow(
                 box(title = "CIRCLE", status="primary",width =
12, solidHeader = TRUE,
                    tableOutput("TableDataOutSmall"),
                    plotOutput("CirclePlotSmall")),
                     box(title=span(icon("fal fa-dna"),"GENOMIC
DATA"), status="primary",width = 12, solidHeader = TRUE,
                    dataTableOutput("GenesSmall"))
            )
```

```
        })
    })


    # Get the data from the plot click
        clickDataSmall <- reactive({unlist(event_data(event =
"plotly_click", source = "smallcircleSource", priority =
"event"))
    })

    # SELECTED CIRCLE INFORMATION
    output$TableDataOutSmall <- renderTable({
        # Import the data file and process it
            circ <-read.table(input$bedfile$datapath,header =
FALSE, sep="\t",stringsAsFactors=FALSE)
                                              names(circ)     <-
c("chrom","start","end","discordant_reads","split_reads","scor
e","coverage_mean","coverage_sd","coverage_start",
"coverage_end","coverage_cont")
        circ$size_bp <- circ$end - circ$start
                                            circ$quality     <-
cut(circ$score,breaks=c(-Inf,10,50,200,Inf),labels=
c("Bad","Low", "Medium", "Good"),right = FALSE)
        circ$chrom <- substr(circ$chrom, start = 1, stop = 5)
        circ$chrom<-str_remove(circ$chrom,"_")
        circ$chrom<-str_remove(circ$chrom,"chr")
                circ$chrom <- factor(circ$chrom, levels =
c("1","2","3","4","5","6","7","8","9","10","11","12","13","14"
,"15","16","17","18","19","20","21","22","Un","M","X","Y"))

        # Use data from the plot to filter
        chrom_filter_small <- unlist(clickDataSmall()[3])
        size_filter_small <- unlist(clickDataSmall()[4])
        sr_filter_small <-unlist(clickDataSmall()[5])

        circ %>%
                  dplyr::filter (chrom==chrom_filter_small &
size_bp==size_filter_small & split_reads==sr_filter_small) %>%

dplyr::select(chrom,start,end,discordant_reads,split_reads,sco
re,coverage_mean,size_bp) %>%

dplyr::rename(Chromosome=chrom,Start=start,End=end,DiscordantR
eads=discordant_reads,SplitReads=split_reads,Score=score,Cover
age=coverage_mean,Size=size_bp)

    })
```

```
    # CIRCULAR PLOT
    output$CirclePlotSmall <-renderPlot({
        # Import the data file and process it
            circ <-read.table(input$bedfile$datapath,header =
FALSE, sep="\t",stringsAsFactors=FALSE)
                                              names(circ)     <-
c("chrom","start","end","discordant_reads","split_reads","scor
e","coverage_mean","coverage_sd","coverage_start",
"coverage_end","coverage_cont")
        circ$size_bp <- circ$end - circ$start
                                             circ$quality     <-
cut(circ$score,breaks=c(-Inf,10,50,200,Inf),labels=
c("Bad","Low", "Medium", "Good"),right = FALSE)
        circ$chrom <- substr(circ$chrom, start = 1, stop = 5)
        circ$chrom<-str_remove(circ$chrom,"_")
        circ$chrom<-str_remove(circ$chrom,"chr")
                circ$chrom  <-  factor(circ$chrom,  levels  =
c("1","2","3","4","5","6","7","8","9","10","11","12","13","14"
,"15","16","17","18","19","20","21","22","Un","M","X","Y"))

        # Use data from the plot to filter
        chrom_filter_small <- unlist(clickDataSmall()[3])
        size_filter_small <- unlist(clickDataSmall()[4])
        sr_filter_small <-unlist(clickDataSmall()[5])

                      selected_circ<-circ   %>%   dplyr::filter
(chrom==chrom_filter_small   &   size_bp==size_filter_small   &
split_reads==sr_filter_small)

        # Use Biostrings to get sequence
                                             my.dnastring     <-
Biostrings::getSeq(BSgenome.Hsapiens.UCSC.hg38,
paste0("chr",selected_circ[,1]),            selected_circ[,2],
selected_circ[,3])
        n<-as.vector(alphabetFrequency(my.dnastring)[1:4])

        # Make a data frame
        base <-c("A","C","G","T")
        data<-data.frame(base,n)
        data$fraction <- data$n / sum(data$n) # Percentages
             data$ymax <- cumsum(data$fraction) # Cumulative
percentages (top of each rectangle)
          data$ymin <- c(0, head(data$ymax, n=-1)) # Bottom of
each rectangle
           data$labelPosition <- (data$ymax + data$ymin) / 2 #
Label position
                       data$label   <-   paste0(data$base,":\n
```

```
",(round((data$fraction)*100,2)),"%") # Label

        # Use data frame to make circular plot
         ggplot(data, aes(ymax=ymax, ymin=ymin, xmax=4, xmin=3,
fill=base)) +
            geom_rect() +
                      geom_label(  x=3.5,  aes(y=labelPosition,
label=label), size=6) +
            scale_fill_brewer(palette=4) +
            coord_polar(theta="y") +
            xlim(c(1, 4)) +
            theme_void() +
            theme(legend.position = "none")
    })

    # TABLE WITH GENES IN CIRCLE
    output$GenesSmall<-  DT::renderDataTable({
        # Import an process the data file
             circ <-read.table(input$bedfile$datapath,header =
FALSE, sep="\t",stringsAsFactors=FALSE)
                                              names(circ)     <-
c("chrom","start","end","discordant_reads","split_reads","scor
e","coverage_mean","coverage_sd","coverage_start",
"coverage_end","coverage_cont")
        circ$size_bp <- circ$end - circ$start
                                             circ$quality     <-
cut(circ$score,breaks=c(-Inf,10,50,200,Inf),labels=
c("Bad","Low", "Medium", "Good"),right = FALSE)
        circ$chrom <- substr(circ$chrom, start = 1, stop = 5)
        circ$chrom<-str_remove(circ$chrom,"_")
        circ$chrom<-str_remove(circ$chrom,"chr")
                  circ$chrom  <-  factor(circ$chrom,  levels  =
c("1","2","3","4","5","6","7","8","9","10","11","12","13","14"
,"15","16","17","18","19","20","21","22","Un","M","X","Y"))

        # Use the data from the previuos plot to filter
        chrom_filter_small <- unlist(clickDataSmall()[3])
        size_filter_small <- unlist(clickDataSmall()[4])
        sr_filter_small <-unlist(clickDataSmall()[5])

                   selected_circ  <-  circ  %>%  dplyr::filter
(chrom==chrom_filter_small   &   size_bp==size_filter_small   &
split_reads==sr_filter_small)

        # Get coordenates

coords<-GRanges(paste0("chr",selected_circ[,1],":",selected_ci
rc[,2],"-",selected_circ[,3]))
```

```
        # Find genes and annotate with gene symbols
        genes <-genes(TxDb.Hsapiens.UCSC.hg38.knownGene)
                                                genes_df      <-
as.data.frame(subsetByOverlaps(genes,coords))
        entrezid <- genes_df$gene_id
        hs <- org.Hs.eg.db
        genesymbol_df<- AnnotationDbi::select(hs,
                                              keys = entrezid,
                                                      columns =
c("ENTREZID", "SYMBOL"),
                                                      keytype =
"ENTREZID")
                  names(genesymbol_df)[names(genesymbol_df)  ==
"ENTREZID"] <- "gene_id"
        genes_smallcircle<-left_join(genes_df,genesymbol_df)

        # Fix names of columns for the table

names(genes_smallcircle)<-c("Chromosome","Start","End","Width"
,"Strand","Gene ID","Gene Symbol")

        # Make a null data frame
            null_gene<- as.data.frame("No genes (or parts of
genes) found in this circle.")

        # Show results
        if(nrow(genes_smallcircle) == 0){
            DT::datatable(null_gene,
                          options = list(
                              searching = FALSE))
        }else{
            DT::datatable(genes_smallcircle,options = list(
                searching = FALSE))
        }
    })


    ## SOURCE: BIG CIRCLES PLOT
     # Change tab and render info when clicking in the plot of
big circles

observeEvent(event_data("plotly_click",source="bigcircleSource
"), {
        circletab2 <- switch(input$tabs,
                             "bigcirc" = "circle")
```

```
        updateTabItems(session, "tabs", circletab2)
    })


observeEvent(event_data("plotly_click",source="bigcircleSource
"), {
        output$selected_circle<- renderUI({
            fluidRow(
                     box(title=span(icon("circle-o"),"CIRCLE"),
status="primary",width = 12, solidHeader = TRUE,
                    tableOutput("TableDataOutBig"),
                    plotOutput("CirclePlotBig")),
                     box(title=span(icon("fal fa-dna"),"GENOMIC
DATA"), status="primary",width = 12, solidHeader = TRUE,
                    dataTableOutput("GenesBig")))

        })
    })

    # Get the data from the plot click
         clickDataBig  <-  reactive({unlist(event_data(event  =
"plotly_click",  source  =  "bigcircleSource",  priority  =
"event"))
    })

    # SELECTED CIRCLE INFORMATION
    output$TableDataOutBig <- renderTable({
        # Import an process the data file
             circ  <-read.table(input$bedfile$datapath,header  =
FALSE, sep="\t",stringsAsFactors=FALSE)
                                              names(circ)     <-
c("chrom","start","end","discordant_reads","split_reads","scor
e","coverage_mean","coverage_sd","coverage_start",
"coverage_end","coverage_cont")
        circ$size_bp <- circ$end - circ$start
                                            circ$quality     <-
cut(circ$score,breaks=c(-Inf,10,50,200,Inf),labels=
c("Bad","Low", "Medium", "Good"),right = FALSE)
        circ$chrom <- substr(circ$chrom, start = 1, stop = 5)
        circ$chrom<-str_remove(circ$chrom,"_")
        circ$chrom<-str_remove(circ$chrom,"chr")
                 circ$chrom  <-  factor(circ$chrom,  levels  =
c("1","2","3","4","5","6","7","8","9","10","11","12","13","14"
,"15","16","17","18","19","20","21","22","Un","M","X","Y"))

        # Use the data from the previuos plot to filter
        chrom_filter_big <- unlist(clickDataBig()[3])
        size_filter_big <- unlist(clickDataBig()[4])
```

```
        sr_filter_big <-unlist(clickDataBig()[5])
        circ %>%
                      dplyr::filter (chrom==chrom_filter_big &
size_bp==size_filter_big & split_reads==sr_filter_big) %>%

dplyr::select(chrom,start,end,discordant_reads,split_reads,sco
re,coverage_mean,size_bp) %>%

dplyr::rename(Chromosome=chrom,Start=start,End=end,DiscordantR
eads=discordant_reads,SplitReads=split_reads,Score=score,Cover
age=coverage_mean,Size=size_bp)
    })

    # CIRCULAR PLOT
    output$CirclePlotBig <-renderPlot({
        # Import an process the data file
            circ <-read.table(input$bedfile$datapath,header =
FALSE, sep="\t",stringsAsFactors=FALSE)
                                              names(circ)     <-
c("chrom","start","end","discordant_reads","split_reads","scor
e","coverage_mean","coverage_sd","coverage_start",
"coverage_end","coverage_cont")
        circ$size_bp <- circ$end - circ$start
                                             circ$quality     <-
cut(circ$score,breaks=c(-Inf,10,50,200,Inf),labels=
c("Bad","Low", "Medium", "Good"),right = FALSE)
        circ$chrom <- substr(circ$chrom, start = 1, stop = 5)
        circ$chrom<-str_remove(circ$chrom,"_")
        circ$chrom<-str_remove(circ$chrom,"chr")
                 circ$chrom  <-  factor(circ$chrom,  levels  =
c("1","2","3","4","5","6","7","8","9","10","11","12","13","14"
,"15","16","17","18","19","20","21","22","Un","M","X","Y"))

        # Use the data from the previuos plot to filter
        chrom_filter_big <- unlist(clickDataBig()[3])
        size_filter_big <- unlist(clickDataBig()[4])
        sr_filter_big <-unlist(clickDataBig()[5])

                      selected_circ<-circ   %>%   dplyr::filter
(chrom==chrom_filter_big    &    size_bp==size_filter_big    &
split_reads==sr_filter_big)

        # Use Biostrings to get sequence
                                             my.dnastring     <-
Biostrings::getSeq(BSgenome.Hsapiens.UCSC.hg38,
paste0("chr",selected_circ[,1]),            selected_circ[,2],
selected_circ[,3])
        n<-as.vector(alphabetFrequency(my.dnastring)[1:4])
```

```
        # Make a data frame
        base <-c("A","C","G","T")
        data<-data.frame(base,n)
        data$fraction <- data$n / sum(data$n) # Percentage
             data$ymax <- cumsum(data$fraction) # Cumulative
percentages (top of each rectangle)
          data$ymin <- c(0, head(data$ymax, n=-1)) # Bottom of
each rectangle
           data$labelPosition <- (data$ymax + data$ymin) / 2 #
Label position
                        data$label   <-   paste0(data$base,":\n
",(round((data$fraction)*100,2)),"%") # Label

        # Use data frame to make circular plot
         ggplot(data, aes(ymax=ymax, ymin=ymin, xmax=4, xmin=3,
fill=base)) +
            geom_rect() +
                      geom_label(  x=3.5,  aes(y=labelPosition,
label=label), size=6) +
            scale_fill_brewer(palette=4) +
            coord_polar(theta="y") +
            xlim(c(1, 4)) +
            theme_void() +
            theme(legend.position = "none")

    })


    #TABLE WITH GENES IN CIRCLE
        output$GenesBig<-  DT::renderDataTable({
            # Import an process the data file
              circ <-read.table(input$bedfile$datapath,header =
FALSE, sep="\t",stringsAsFactors=FALSE)
                                                 names(circ)    <-
c("chrom","start","end","discordant_reads","split_reads","scor
e","coverage_mean","coverage_sd","coverage_start",
"coverage_end","coverage_cont")
            circ$size_bp <- circ$end - circ$start
                                               circ$quality    <-
cut(circ$score,breaks=c(-Inf,10,50,200,Inf),labels=
c("Bad","Low", "Medium", "Good"),right = FALSE)
             circ$chrom <- substr(circ$chrom, start = 1, stop =
5)
            circ$chrom<-str_remove(circ$chrom,"_")
            circ$chrom<-str_remove(circ$chrom,"chr")
                   circ$chrom <- factor(circ$chrom, levels =
c("1","2","3","4","5","6","7","8","9","10","11","12","13","14"
,"15","16","17","18","19","20","21","22","Un","M","X","Y"))
```

```
            # Use the data from the previuos plot to filter
            chrom_filter_big <- unlist(clickDataBig()[3])
            size_filter_big <- unlist(clickDataBig()[4])
            sr_filter_big <-unlist(clickDataBig()[5])

                    selected_circ <- circ %>% dplyr::filter
(chrom==chrom_filter_big    &    size_bp==size_filter_big    &
split_reads==sr_filter_big)

            # Get coordenates

coords<-GRanges(paste0("chr",selected_circ[,1],":",selected_ci
rc[,2],"-",selected_circ[,3]))

            # Find genes and annotate with gene symbols
            genes <-genes(TxDb.Hsapiens.UCSC.hg38.knownGene)
                                                genes_df    <-
as.data.frame(subsetByOverlaps(genes,coords))
            entrezid <- genes_df$gene_id
            hs <- org.Hs.eg.db
            genesymbol_df<- AnnotationDbi::select(hs,
                                                        keys =
entrezid,
                                                    columns =
c("ENTREZID", "SYMBOL"),
                                                    keytype =
"ENTREZID")
                  names(genesymbol_df)[names(genesymbol_df)  ==
"ENTREZID"] <- "gene_id"
            genes_bigcircle<-left_join(genes_df,genesymbol_df)

            # Fix names of columns for the table

names(genes_bigcircle)<-c("Chromosome","Start","End","Width","
Strand","Gene ID","Gene Symbol")

            # Make a null data frame
              null_gene<- as.data.frame("No genes (or parts of
genes) found in this circle.")

            # Show results
           if(nrow(genes_bigcircle) == 0){
               DT::datatable(null_gene,
                             options = list(
                   searching = FALSE))
            }else{
```

```
                DT::datatable(genes_bigcircle,options = list(
                    searching = FALSE))
                }

        })

}

shinyApp(ui=ui, server = server)
```

Dockerfile

```
FROM rocker/shiny:4.1.0
# system libraries of general use
RUN apt-get update && apt-get install -y \
    sudo \
    pandoc \
    pandoc-citeproc \
    libcurl4-gnutls-dev \
    libcairo2-dev \
    libxt-dev \
    libssl-dev \
    libssh2-1-dev \
    libxml2 \
    libxml2-dev \
    libv8-dev libnode-dev \
    r-cran-xml \
    r-cran-igraph \
    r-cran-rglpk r-cran-rstan

## rm -rf /var/lib/apt/lists/*
# install R packages required
RUN R -e "install.packages('devtools', repos='http://cran.rstudio.com/')"
RUN R -e "install.packages('BiocManager', repos='http://cran.rstudio.com/', version = "3.13")"

RUN R -e "install.packages('rstan', repos='http://cran.rstudio.com/')"
# bioC 3.13
# RUN R -e "BiocManager::install(version = '3.13', ask = FALSE)"
RUN R -e "devtools::install_version('shiny', version = '1.6.0', repos='http://cran.rstudio.com/')"
RUN R -e "devtools::install_version('shinydashboard', version = '0.7.1', repos='http://cran.rstudio.com/')"
```

```
RUN R -e "devtools::install_version('dplyr', version =
'1.0.6', repos='http://cran.rstudio.com/')"
RUN R -e "devtools::install_version('ggplot2', version =
'3.3.3', repos='http://cran.rstudio.com/')"
RUN R -e "devtools::install_version('stringr', version =
'1.4.0', repos='http://cran.rstudio.com/')"
RUN R -e "devtools::install_version('plotly', version =
'4.9.3', repos='http://cran.rstudio.com/')"
RUN R -e "devtools::install_version('DT', version = '0.18',
repos='http://cran.rstudio.com/')"
RUN R -e
"BiocManager::install(c('TxDb.Hsapiens.UCSC.hg38.knownGene',
'GenomicRanges', 'org.Hs.eg.db', 'RITANdata',
'Biostrings','BSgenome.Hsapiens.UCSC.hg38'), version =
'3.13')"
RUN R -e "BiocManager::install('STRINGdb')"
RUN R -e "BiocManager::install(c('RITAN'), version = '3.13')"

COPY visualeccdna ./app


# select port
EXPOSE 3838

CMD ["R", "-e", "shiny::runApp('/app', host = '0.0.0.0', port
= 3838)"]
```